CS 2740 Knowledge representation
Lecture 21

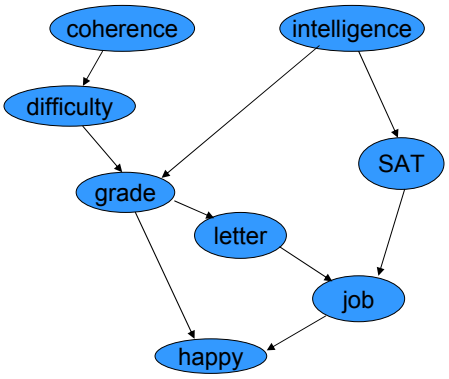
Bayesian belief networks: Inference

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Variable elimination

The order in which variables are eliminated may effect the efficiency of the variable elimination process

Assume the following BBN and calculation of *P(Job)*:



Variable elimination

Calculations performed in terms of factors:

$$p(J) = \sum_{L,S,G,H,I,D,C} \phi_C(c)\phi_I(i)\phi_D(d,c)\phi_G(g,i,d)\phi_S(s,i)\phi_L(l,g)\phi_J(j,l,s)\phi_H(h,g,j)$$

$$= \sum_{L,S,G,H,I,D} \phi_I(i)\phi_G(g,i,d)\phi_S(s,i)\phi_L(l,g)\phi_J(j,l,s)\phi_H(h,g,j)\sum_C \phi_C(c)\phi_D(d,c)$$

$$= \sum_{L,S,G,H,I,D} \phi_I(i)\phi_G(g,i,d)\phi_S(s,i)\phi_L(l,g)\phi_J(j,l,s)\phi_H(h,g,j)\tau_1(d)$$

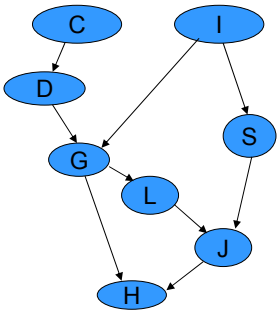
...

$$= \sum_{L,S} \phi_J(j,l,s)\sum_G \phi_L(l,g)\tau(s,g)\tau(g,j)$$

$$= \sum_{L,S} \phi_J(j,l,s)\tau(l,s,j)$$

$$= \sum_L \tau(l,j)$$

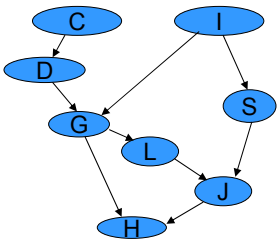
$$= \tau(j)$$



Variable elimination

Trace 1:

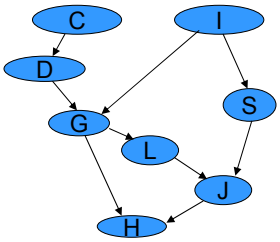
Step	Var	Factors Used	New Factor
1	C	$\phi_c(C), \phi_D(D,C)$	$\tau_1(D)$
2	D	$\phi_G(G,I,D), \tau_1(D)$	$\tau_2(G,I)$
3	I	$\phi_I(I), \phi_S(S,I), \tau_2(G,I)$	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	$\tau_4(G,J)$
5	G	$\tau_4(G,J), \tau_3(G,S), \phi_L(L,G)$	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S), \phi_J(J,L,S)$	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	$\tau_7(J)$



Variable elimination

Trace 1:

Step	Var	Factors Used	New Factor
1	C	$\phi_c(C), \phi_D(D,C)$	$\tau_1(D)$
2	D	$\phi_G(G,I,D), \tau_1(D)$	$\tau_2(G,I)$
3	I	$\phi_I(I), \phi_S(S,I), \tau_2(G,I)$	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	$\tau_4(G,J)$
5	G	$\tau_4(G,J), \tau_3(G,S), \phi_L(L,G)$	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S), \phi_J(J,L,S)$	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	$\tau_7(J)$

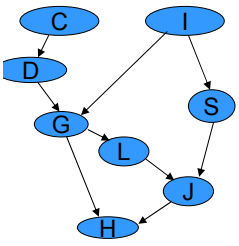


Complexity: 4 variables – 1 summed away

Variable elimination

Trace 2:

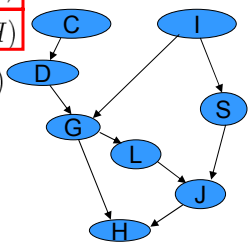
Step	Var	Factors Used	New Factor
1	G	$\phi_G(G,I,D), \phi_L(L,G)\phi_H(H,G,J)$	$\tau_1(I,D,L,J,H)$
2	I	$\phi_I(I), \phi_S(S,I)\tau_1(I,D,L,J,H)$	$\tau_2(D,L,S,J,H)$
3	S	$\phi_J(J,L,S), \tau_2(D,L,S,J,H)$	$\tau_3(D,L,J,H)$
4	L	$\tau_3(D,L,J,H)$	$\tau_4(D,J,H)$
5	H	$\tau_4(D,J,H)$	$\tau_5(D,J)$
6	C	$\tau_5(D,J), \phi_D(D,C)$	$\tau_6(D,J)$
7	D	$\tau_6(D,J)$	$\tau_7(J)$



Variable elimination

Trace 2:

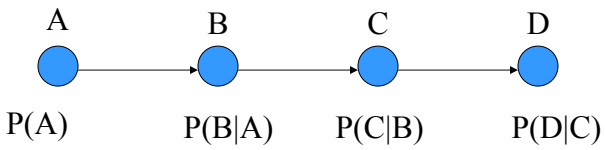
Step	Var	Factors Used	New Factor
1	G	$\phi_G(G,I,D), \phi_L(L,G)\phi_H(H,G,J)$	$\tau_1(I,D,L,J,H)$
2	I	$\phi_I(I), \phi_S(S,I)\tau_1(I,D,L,J,H)$	$\tau_2(D,L,S,J,H)$
3	S	$\phi_J(J,L,S), \tau_2(D,L,S,J,H)$	$\tau_3(D,L,J,H)$
4	L	$\tau_3(D,L,J,H)$	$\tau_4(D,J,H)$
5	H	$\tau_4(D,J,H)$	$\tau_5(D,J)$
6	C	$\tau_5(D,J), \phi_D(D,C)$	$\tau_6(D,J)$
7	D	$\tau_6(D,J)$	$\tau_7(J)$



Complexity: 6 variables used – 1 summed out

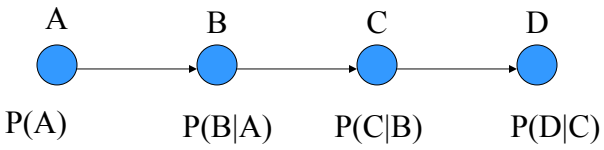
Efficient inference

- Is there a BBN structure that support efficient inference?
- **Yes: Tree**



Efficient inference

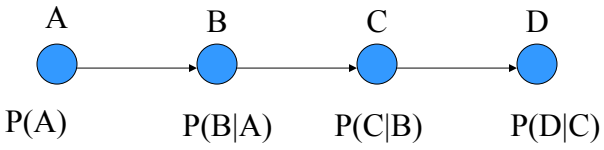
- Is there a BBN structure that support efficient inference?
- **Yes: Tree**



$$P(A=T,B=F,C=T,D=T)=$$
$$P(A=T)P(B=F\mid A=T)P(C=T\mid B=F)P(D=T\mid C=T)$$

Efficient inference

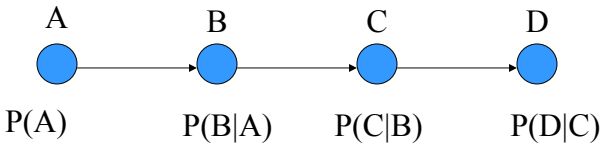
- Is there a BBN structure that support efficient inference?
- **Yes: Tree**



$$P(D=T)=$$
$$=\sum_{a,b,c} P(A=a)P(B=b\mid A=a)P(C=c\mid B=b)P(D=T\mid C=c)$$
$$=\sum_{b,c} P(C=c\mid B=b)P(D=T\mid C=c)\sum_{a} P(A=a)P(B=b\mid A=a)$$
$$=\sum_{b,c} P(C=c\mid B=b)P(D=T\mid C=c)\tau_1(B)$$
$$=\sum_{c} P(D=T\mid C=c)\sum_{b} \tau_1(B)P(C=c\mid B=b)$$
$$=\sum_{c} P(D=T\mid C=c)\tau_2(C=c)$$
$$=\tau_3$$

Efficient inference

- Is there a BBN structure that support efficient inference?
- **Yes: Tree**



Efficient elimination order exists:
Size of the largest factor is 2

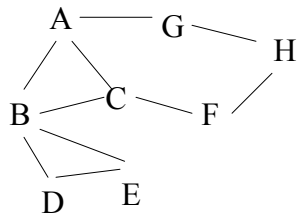
Tree supports efficient inference

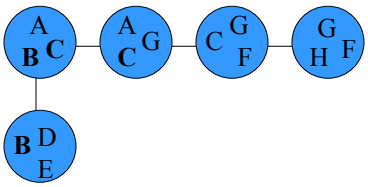
BBN → tree conversion

- Is it possible to transform any BBN to a tree structure that support efficient inference?
- **Yes**
- **The structure is refered to as a joint tree or a tree decomposition of the BBN graph**

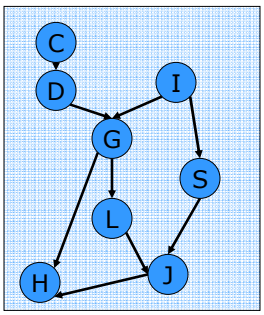
Tree decomposition of the graph

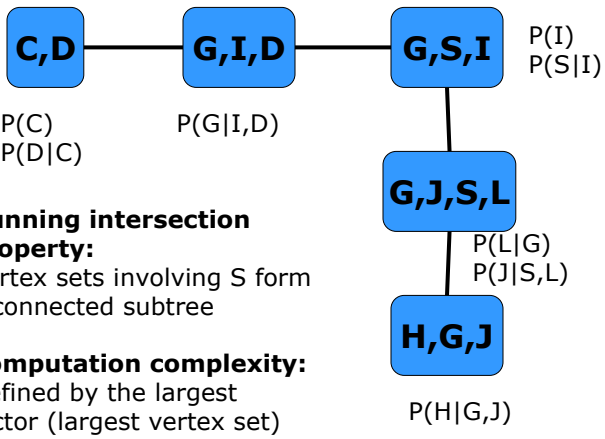
- **A tree decomposition of a graph G:**
  - A tree T with a vertex set associated to every node.
  - For all edges $\{v,w\} \in G$: there is a set containing both v and w in T.
  - For every $v \in G$: the nodes in T that contain v form a connected subtree.





Tree decomposition





Running intersection property:
Vertex sets involving S form a connected subtree

Computation complexity:
Defined by the largest factor (largest vertex set)

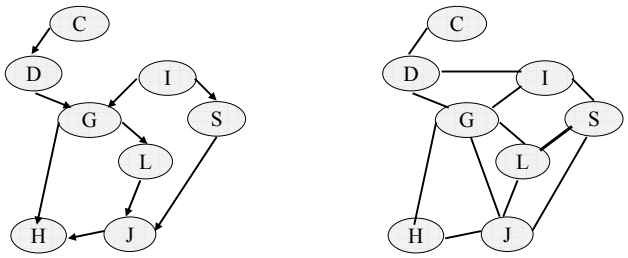
BBN → tree conversion

- **Approaches:**
  (1) moral graphs + triangulations
  (2) a trace of the variable elimination procedure

Moral graph

Moral-graph of a BBN over X is an undirected graph over X that contains an edge between x and y if:

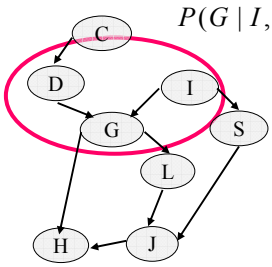
- There exists a directed edge between them in G.
- They are both parents of the same node in G.

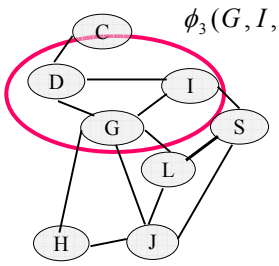


Moral Graphs

Why moralization?

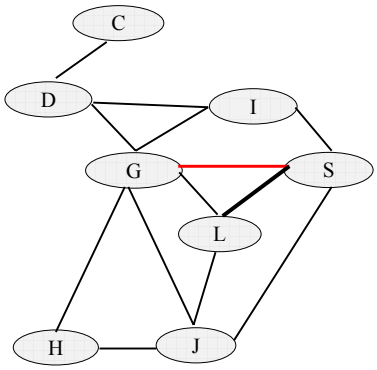
$$P(C,D,G,I,S,L,J,H) =$$
$$= P(C)P(D \mid C)P(G \mid I,D)P(S \mid I)P(L \mid G)P(J \mid L,S)P(H \mid G,J)$$
$$= \phi_1(C)\phi_2(D,C)\phi_3(G,I,D)\phi_4(S,I)\phi_5(L,G)\phi_6(J,L,S)\phi_7(H,G,J)$$





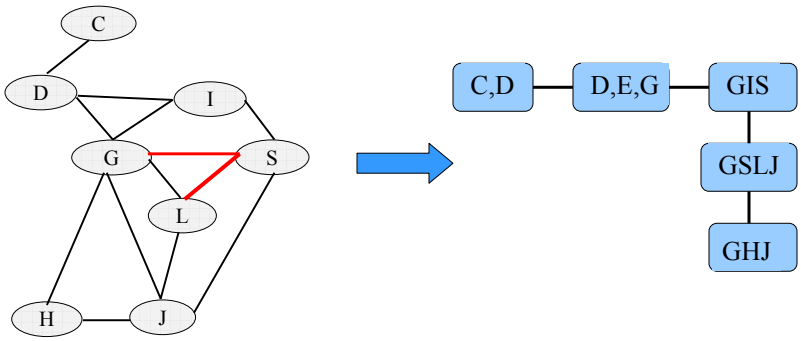
Triangulation

- Break all cycles of length > 3 with an undirected link
- **Note:**
  - Can be tricky
  - More than one solution



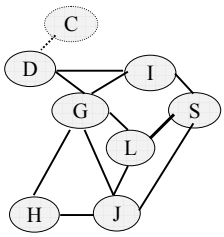
Tree decomposition

- Tree is obtained by reading out graph cliques and representing each clique by a vertex set in the tree graph
- Tree is refereed to as: **a joint tree** or **a clique tree**



VE: Trace (1)

(1)



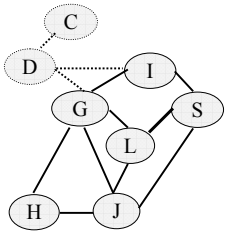
a) Multiply the factors to produce:

$$\phi(D,C) = \phi(D) \times \phi(C)$$

b) Sum over C to produce:

$$\tau(D) = \sum_{C} \phi(D,C)$$

(2)



a) Multiply the factors to produce:

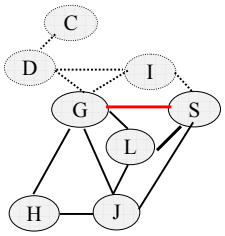
$$\phi(D,I,G) = \phi(G,I,D) \times \tau_1(D)$$

b) Sum over D to produce:

$$\tau_2(G,I) = \sum_{D} \phi(D,I,G)$$

VE: Trace (2)

(3)



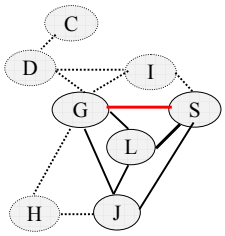
a) Multiply the factors to produce:

$$\phi(I,G,S) = \phi(I) \times \phi(S,I) \times \tau_2(G,I)$$

b) Sum over I to produce:

$$\tau_3(G,S) = \sum_I \phi(I,G,S)$$

(4)



a) Multiply the factors to produce:

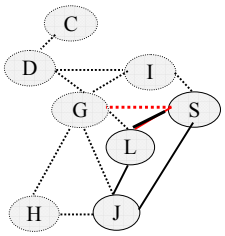
$$\phi(H,G,J) = \phi(H,G,J)$$

b) Sum over I to produce:

$$\tau_4(G,J) = \sum_H \phi(H,G,J)$$

VE: Trace (3)

(5)



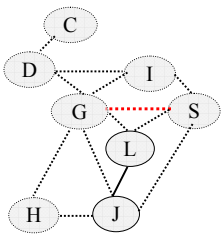
a) Multiply the factors to produce:

$$\phi(G,J,L,S) = \tau_4(G,J) \times \tau_3(G,S) \times \phi(L,G$$

b) Sum over I to produce:

$$\tau_5(J,L,S) = \sum_G \phi(G,J,L,S)$$

(6)



a) Multiply the factors to produce:

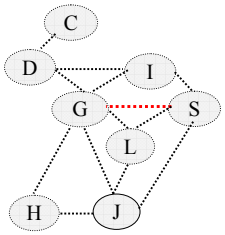
$$\phi(J,L,S) = \tau_5(J,L,S) \times \phi(J,L,S)$$

b) Sum over I to produce:

$$\tau_6(J,L) = \sum_S \phi(J,L,S)$$

VE: Trace (4)

(5)



a) Multiply the factors to produce:

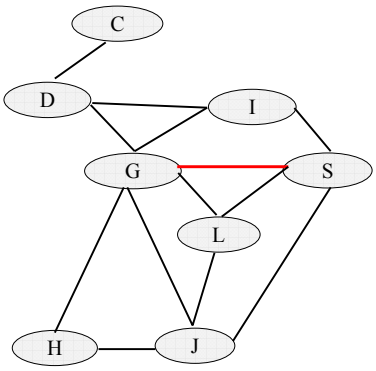
$$\phi(J,L) = \tau_6(J,L)$$

b) Sum over I to produce:

$$\tau_7(J) = \sum_L \phi(J,L)$$

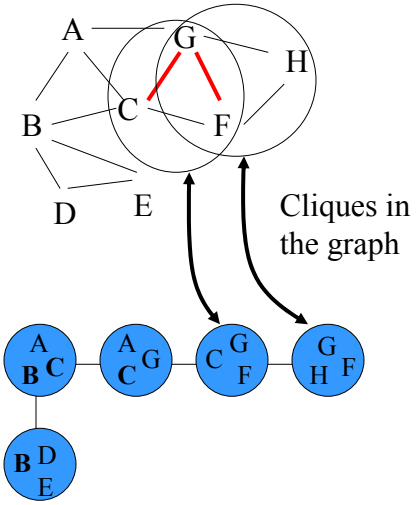
Variable elimination: Induced Graph

- Graph induced by the VE algorithm is a Triangulated graph
- A tree (joint tree) can be constructed from the graph



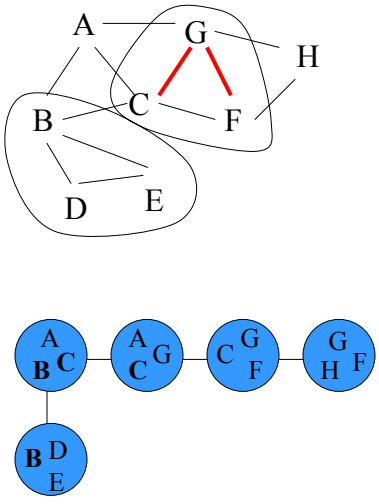
Tree decomposition of the graph

- **A tree decomposition of a graph G:**
  - A tree T with a vertex set associated to every node.
  - For all edges $\{v,w\} \in G$: there is a set containing both v and w in T.
  - For every $v \in G$: the nodes in T that contain v form a connected subtree.



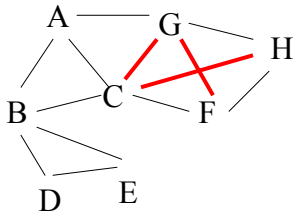
Tree decomposition of the graph

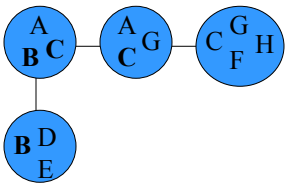
- **A tree decomposition of a graph G:**
  - A tree T with a vertex set associated to every node.
  - For all edges $\{v,w\} \in G$: there is a set containing both v and w in T.
  - For every $v \in G$: the nodes in T that contain v form a connected subtree.



Tree decomposition of the graph

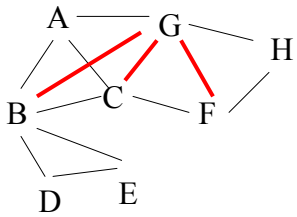
- **Another tree decomposition of a graph G:**
  - A tree T with a vertex set associated to every node.
  - For all edges $\{v,w\} \in G$: there is a set containing both v and w in T.
  - For every $v \in G$: the nodes in T that contain v form a connected subtree.

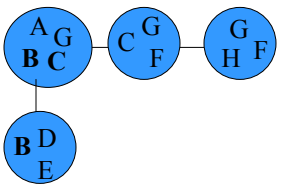




Tree decomposition of the graph

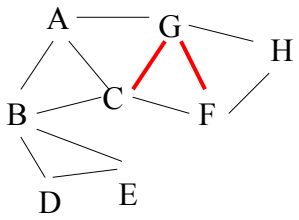
- **Another tree decomposition of a graph G:**
  - A tree T with a vertex set associated to every node.
  - For all edges $\{v,w\} \in G$: there is a set containing both v and w in T.
  - For every $v \in G$: the nodes in T that contain v form a connected subtree.

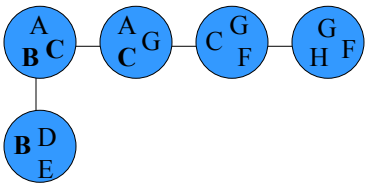




Treewidth of the graph

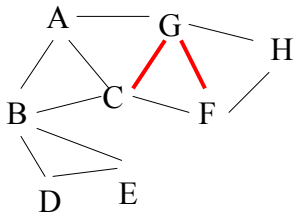
- **Width** of the tree decomposition:
  $\max_{i \in I} |X_i| - 1$
- **Treewidth** of a graph G: tw(G)= minimum width over all tree decompositions of G.

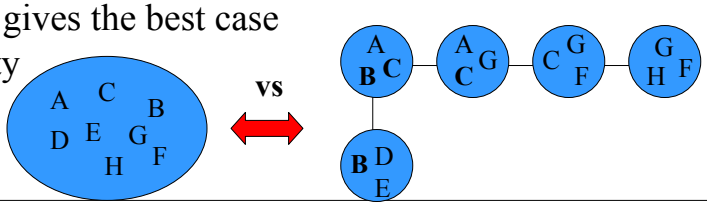




Treewidth of the graph

- **Treewidth** of a graph G: tw(G)= minimum width over all tree decompositions of G
- Why is it important?
- The calculations can take advantage of the structure and be performed more efficiently
- treewidth gives the best case complexity





Inference in Bayesian network

- **Exact inference algorithms:**
  - **Variable elimination**
  - Recursive decomposition (Cooper, Darwiche)
  - Belief propagation algorithm (Pearl)
  - Arc reversal (Olmsted, Schachter)

- **Approximate inference algorithms:**
  - **Monte Carlo methods:**
    - Forward sampling, Likelihood sampling
  - Variational methods

Monte Carlo approaches

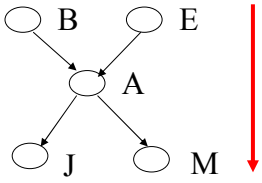
- **MC approximation**:
  - The probability is approximated using sample frequencies
  - **Example:**

$$\tilde{P}(B=T, J=T) = \frac{N_{B=T,J=T}}{N}$$

$N_{B=T,J=T}$ ← *# examples with* $B=T, J=T$

N ← *total # examples*

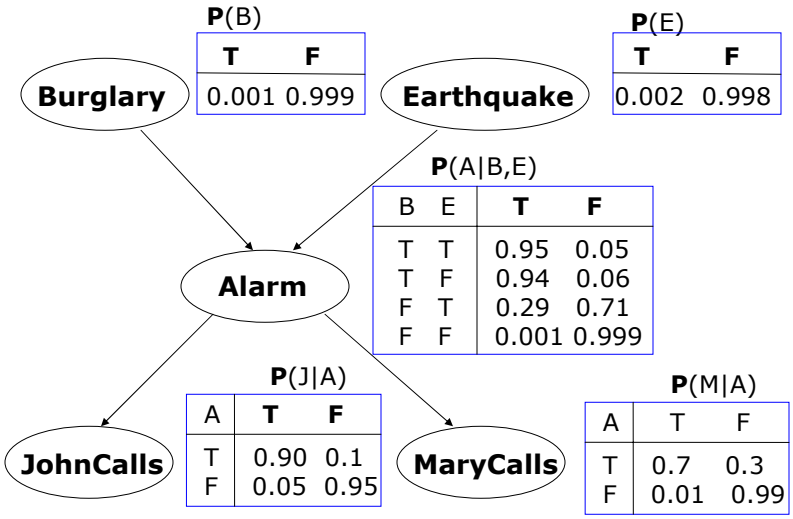
- **BBN sampling:**



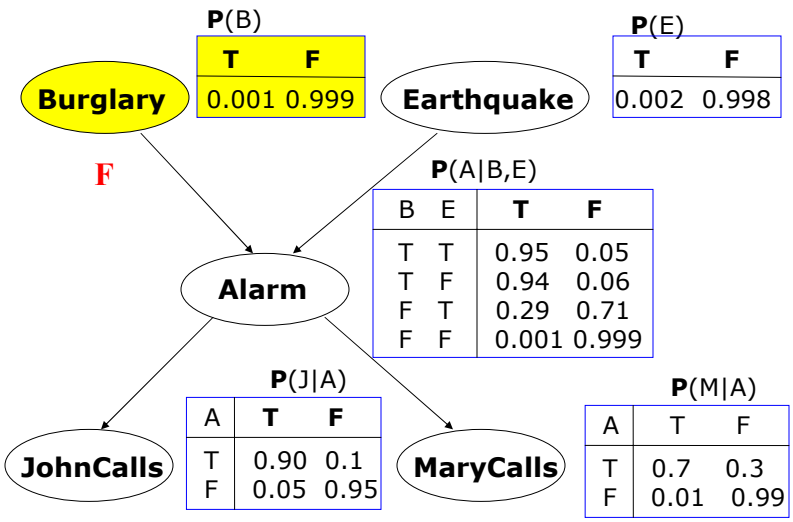
Generate sample in the top down manner, following the links

- **<u>One example gives one assignment of values to all variables</u>**

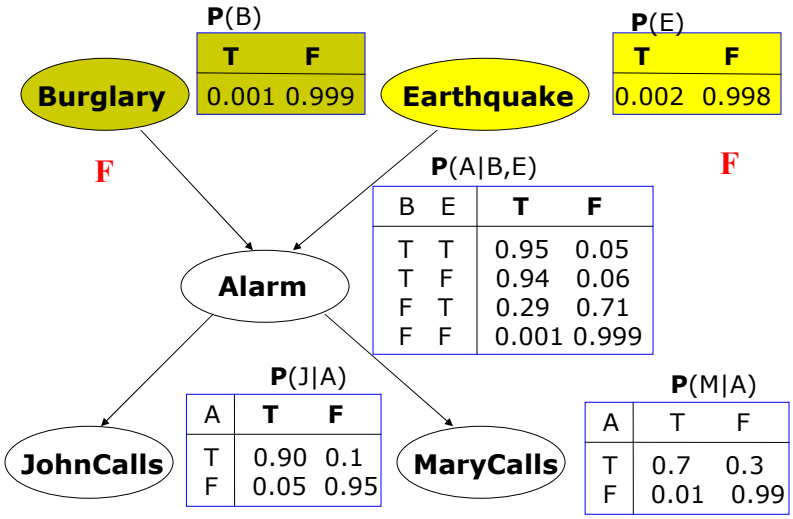
BBN sampling example



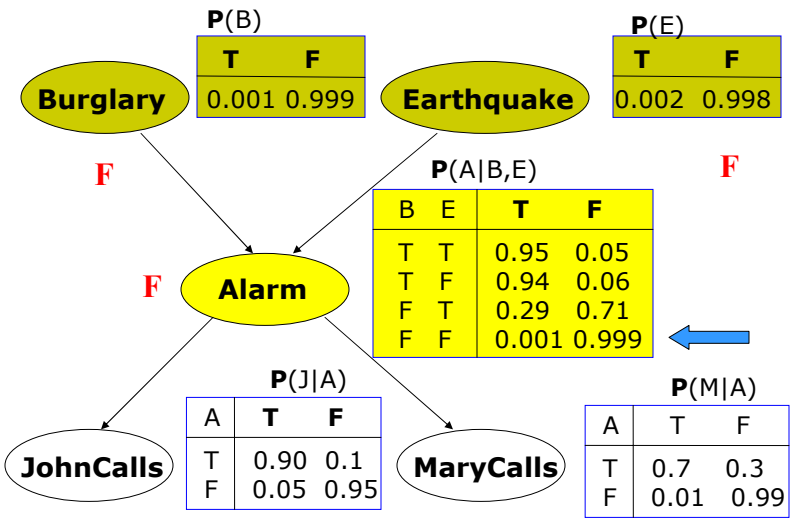
BBN sampling example



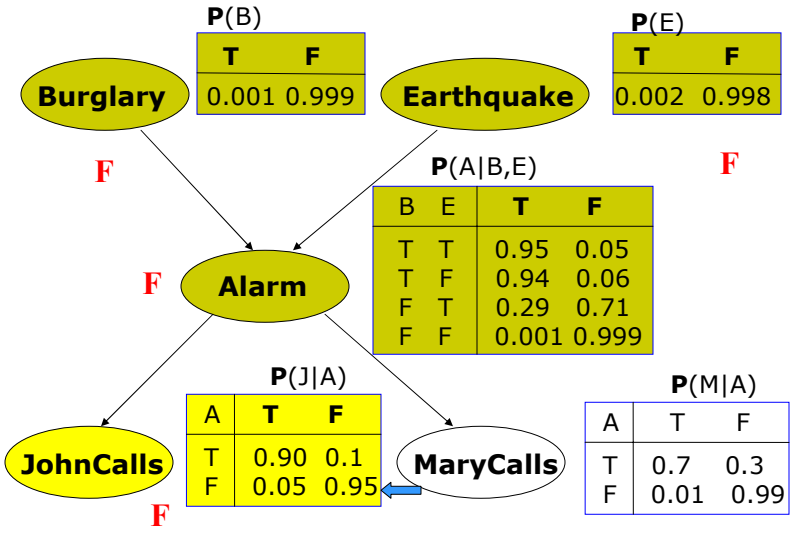
BBN sampling example



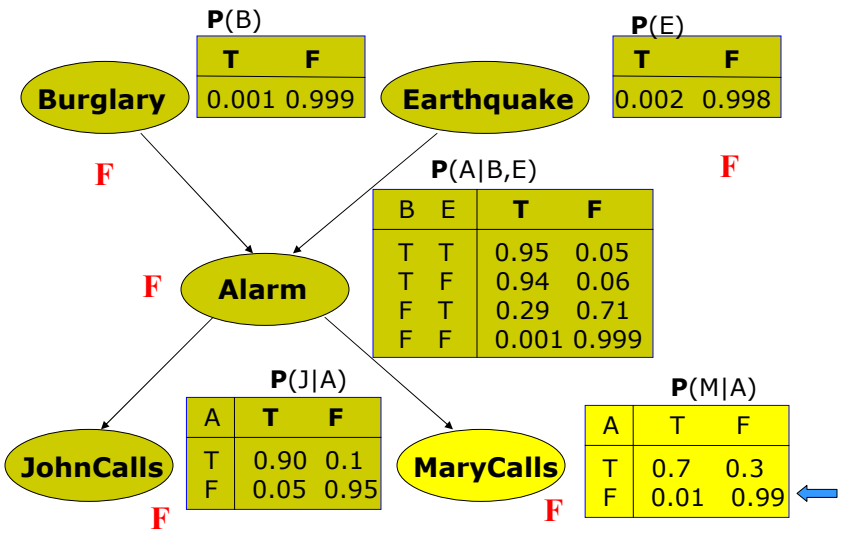
BBN sampling example



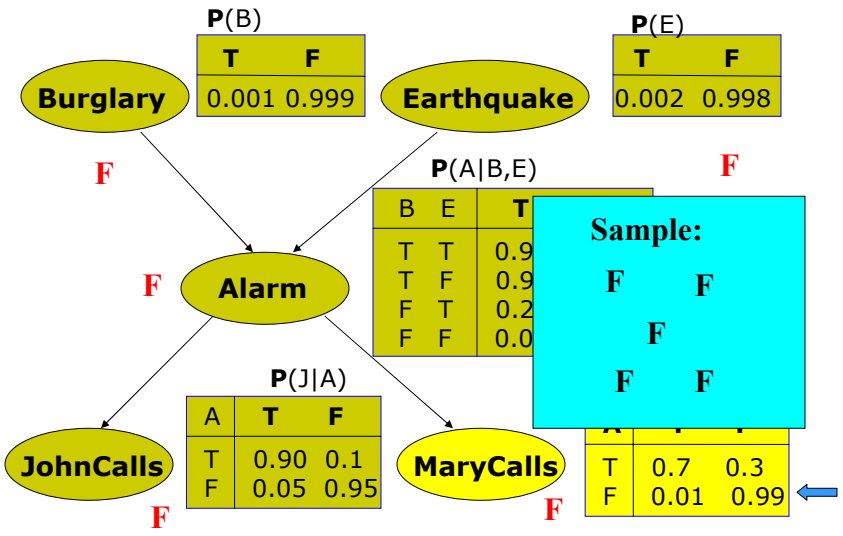
BBN sampling example



BBN sampling example



BBN sampling example



Monte Carlo approaches

- **MC approximation of conditional probabilities**:
  - The probability is approximated using sample frequencies
  - **Example:**

$$\tilde{P}(B=T \mid J=T) = \frac{N_{B=T,J=T}}{N_{J=T}}$$

$N_{B=T,J=T}$: # *samples with* $B=T, J=T$

$N_{J=T}$: # *samples with* $J=T$

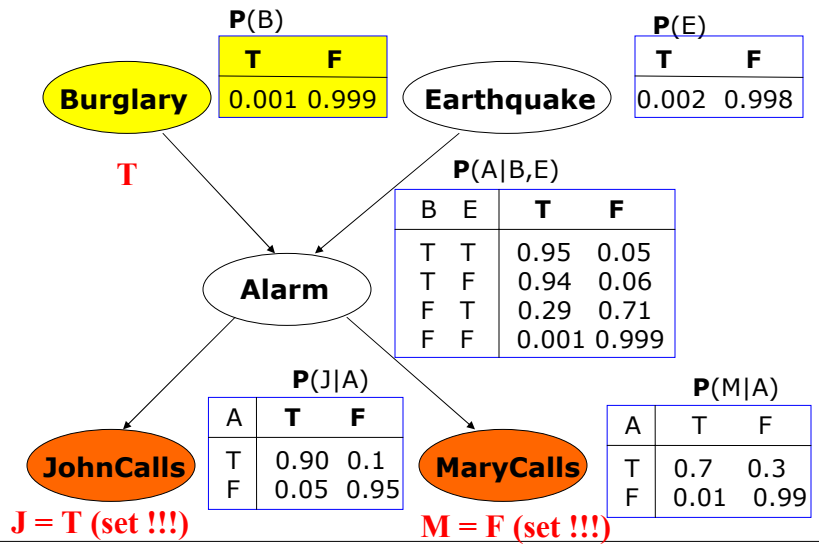
- **Rejection sampling:**
  - Generate samples from the full joint by sampling BBN
  - Use only samples that agree with the condition, the remaining samples are rejected
- **Problem:** many samples can be rejected

Likelihood weighting

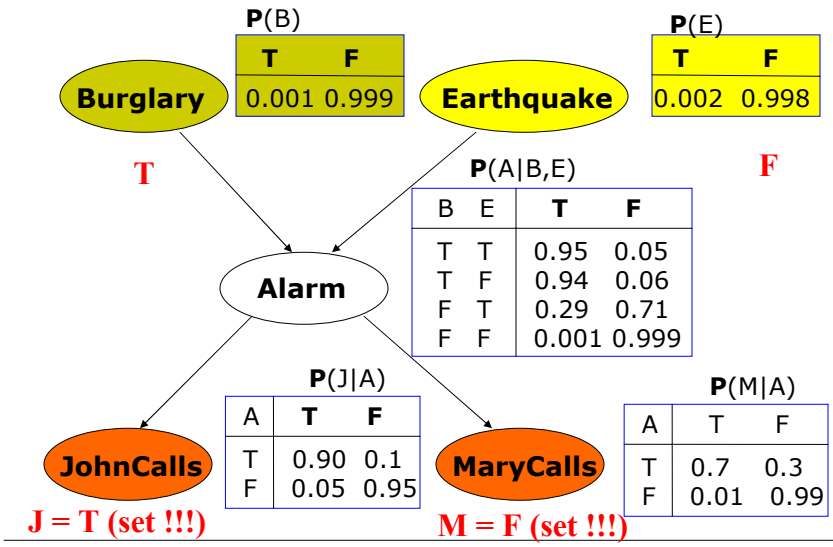
- **Avoids inefficiencies of rejection sampling**
  - **Idea:** generate only samples **consistent with the evidence** (or conditioning event)
    - **the value of evidence nodes is not sampled**
- **Problem:** using simple counts is not enough since these may occur with different probabilities
- Likelihood weighting:
  - **With every sample keep a weight with which it should count towards the estimate**

$$\widetilde{P}(B=T \mid J=T) = \frac{\sum_{\text{samples with } B=T \text{ and } J=T} w_{B=T}}{\sum_{\text{samples with any value of } B \text{ and } J=T} w_{B=x}}$$

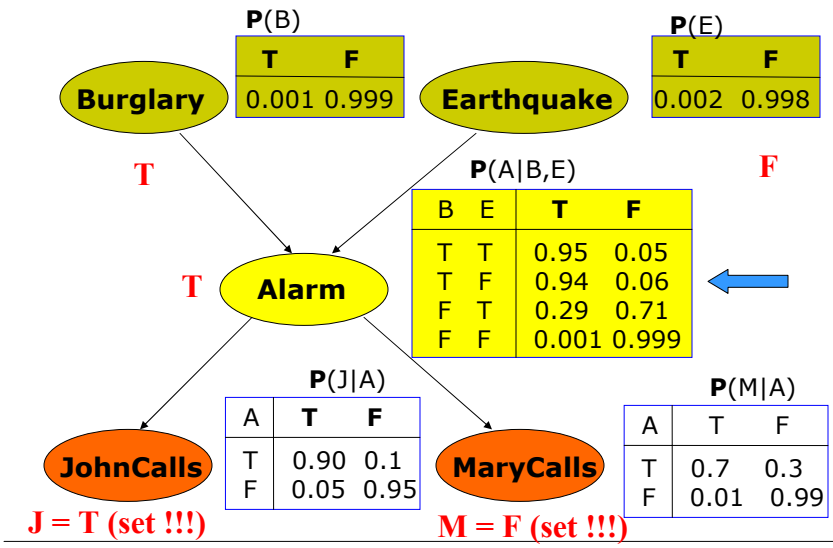
BBN likelihood weighting example



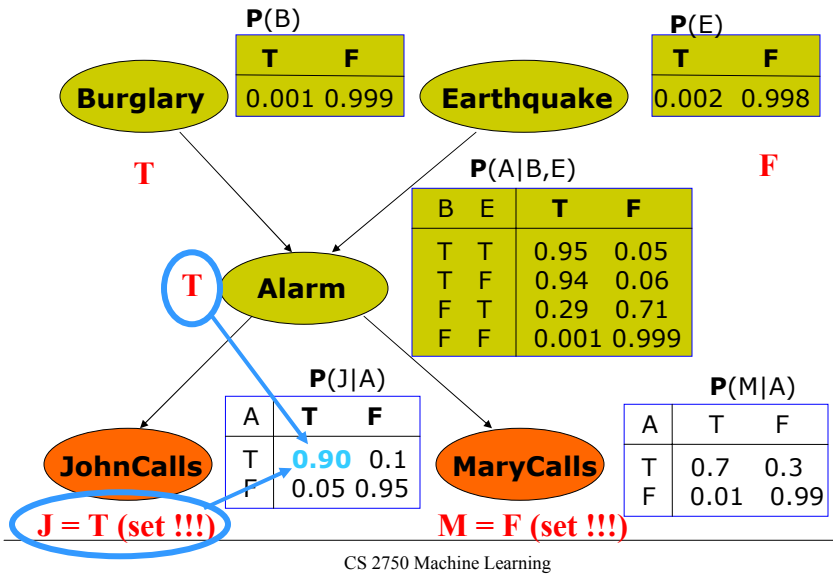
BBN likelihood weighting example



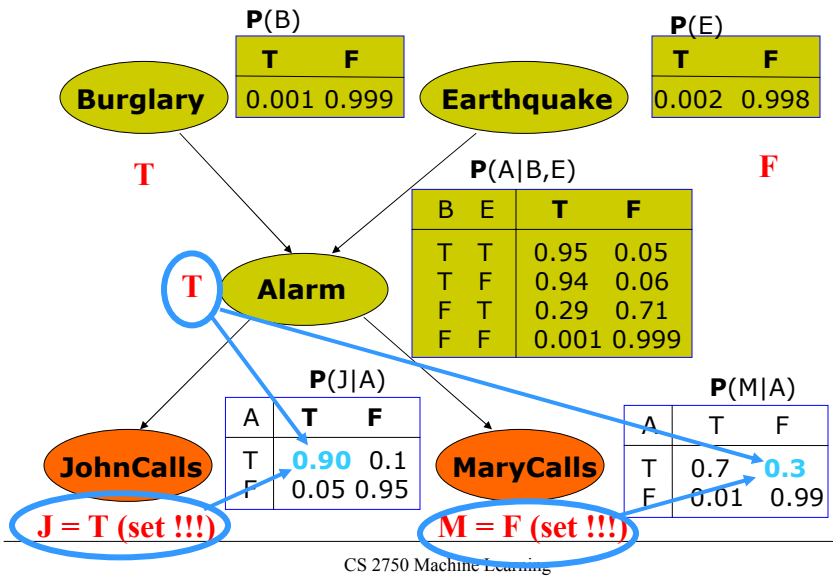
BBN likelihood weighting example



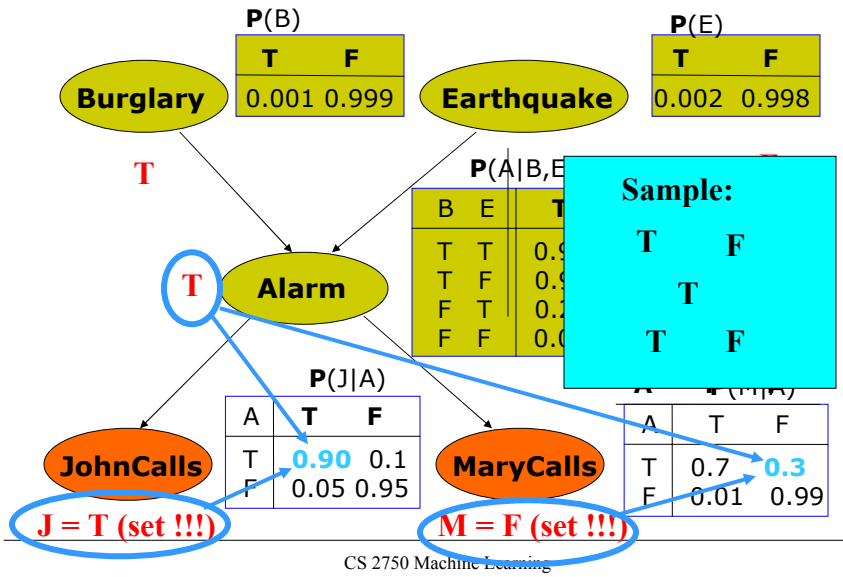
BBN likelihood weighting example



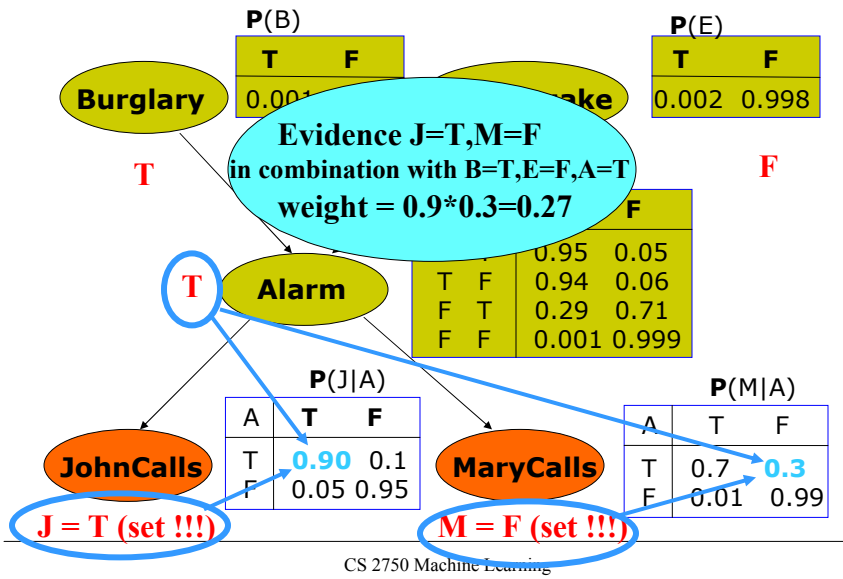
BBN likelihood weighting example



BBN likelihood weighting example

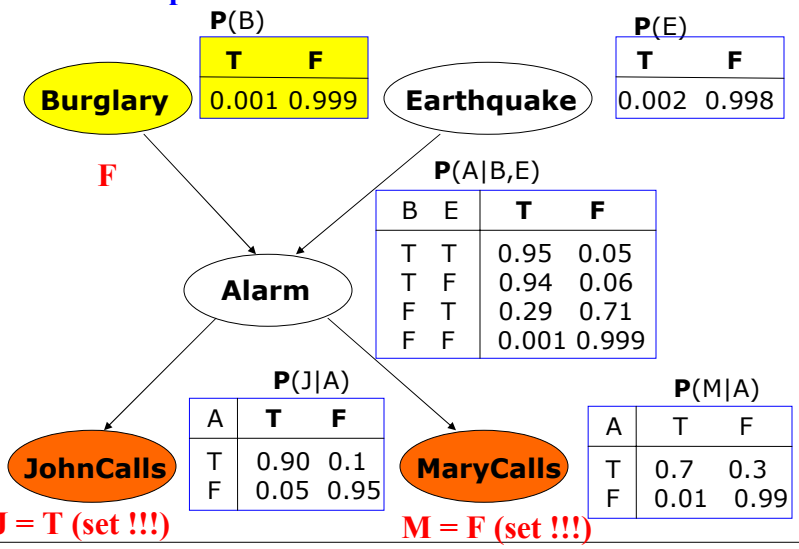


BBN likelihood weighting example



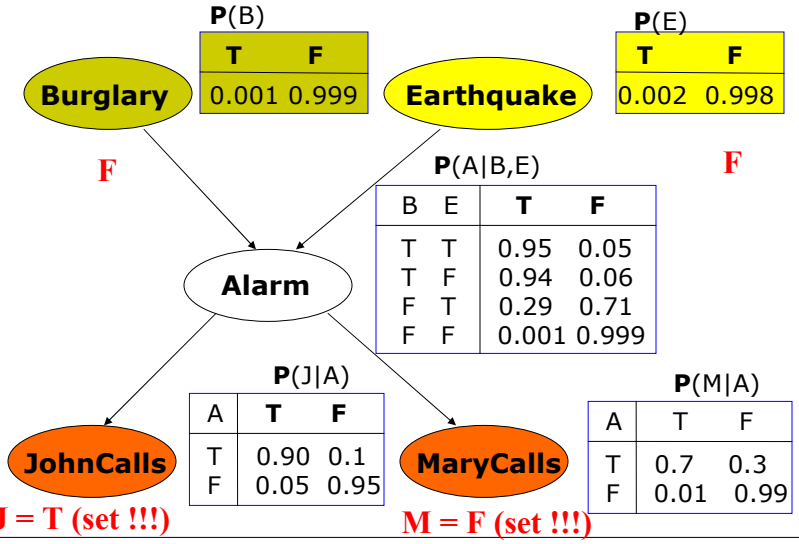
BBN likelihood weighting example

Second sample



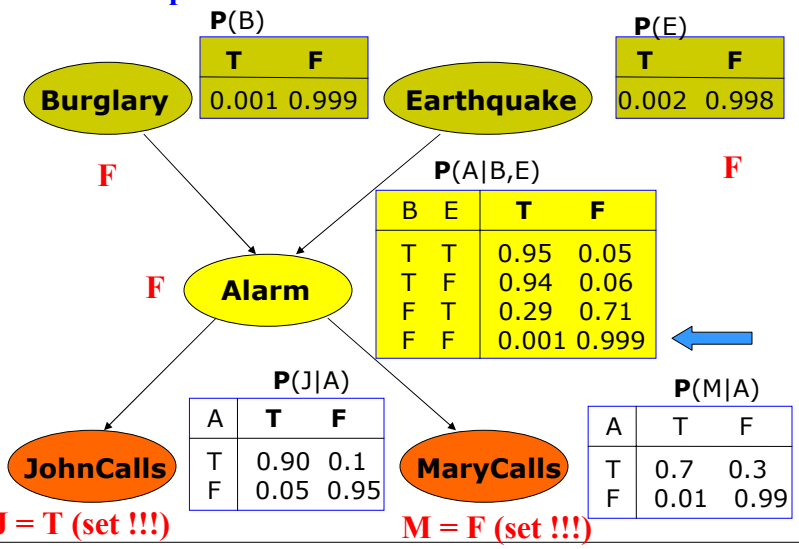
BBN likelihood weighting example

Second sample



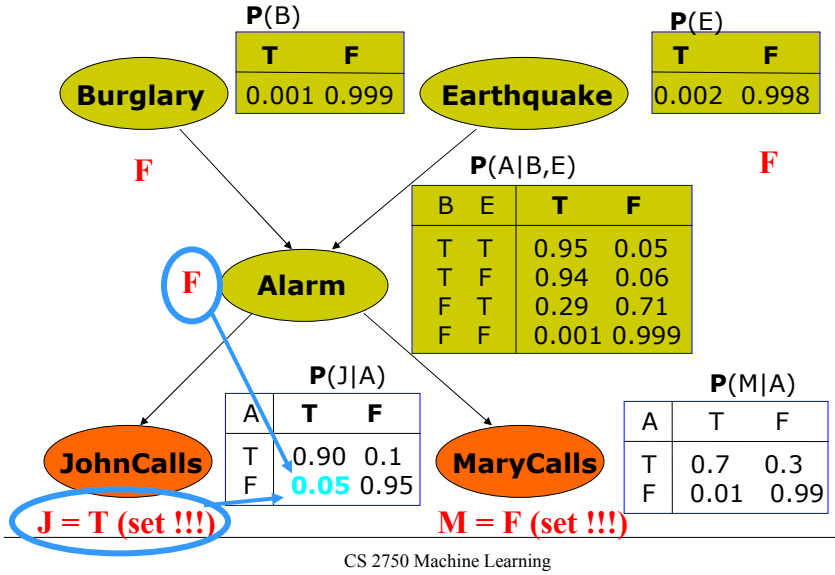
BBN likelihood weighting example

Second sample



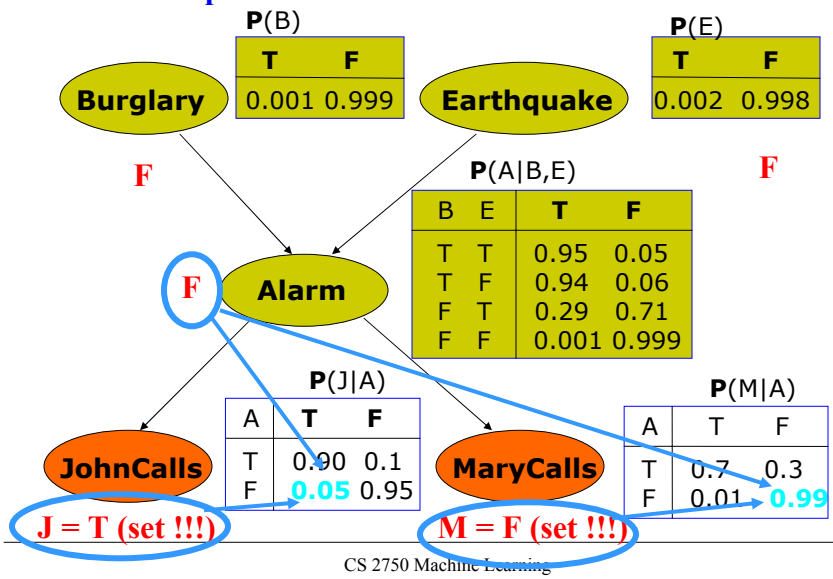
BBN likelihood weighting example

Second sample



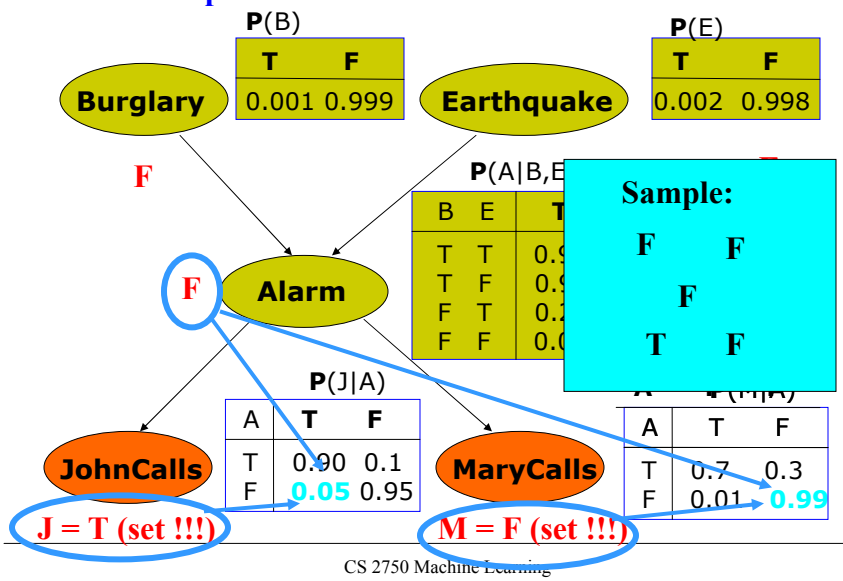
BBN likelihood weighting example

Second sample



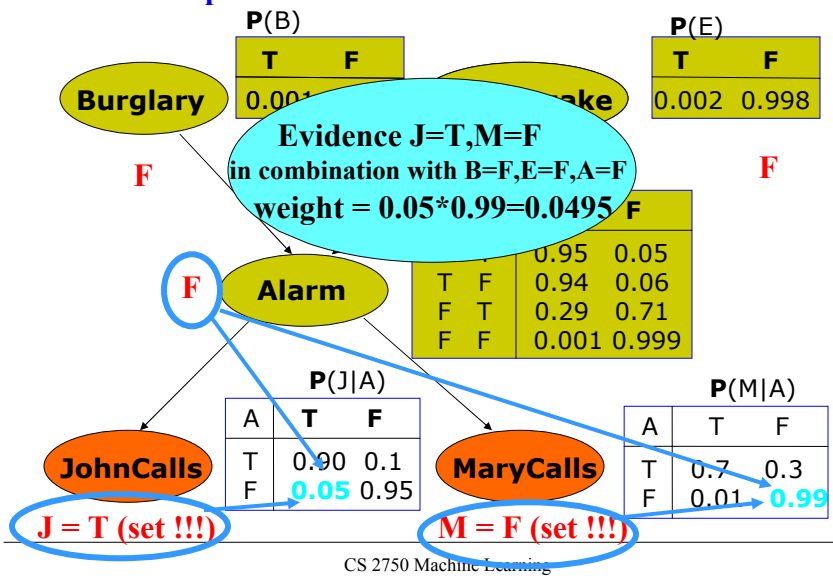
BBN likelihood weighting example

Second sample



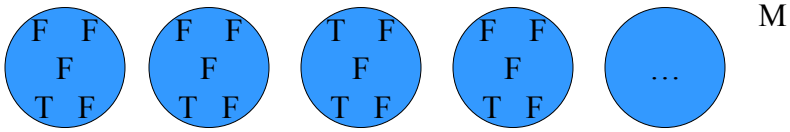
BBN likelihood weighting example

Second sample



Likelihood weighting

- Assume we have generated the following M samples:



How to make examples consistent with the original distribution?

Weight each sample by probability with which it agrees with the conditioning evidence P(e).