Inference Using Bayes Theorem

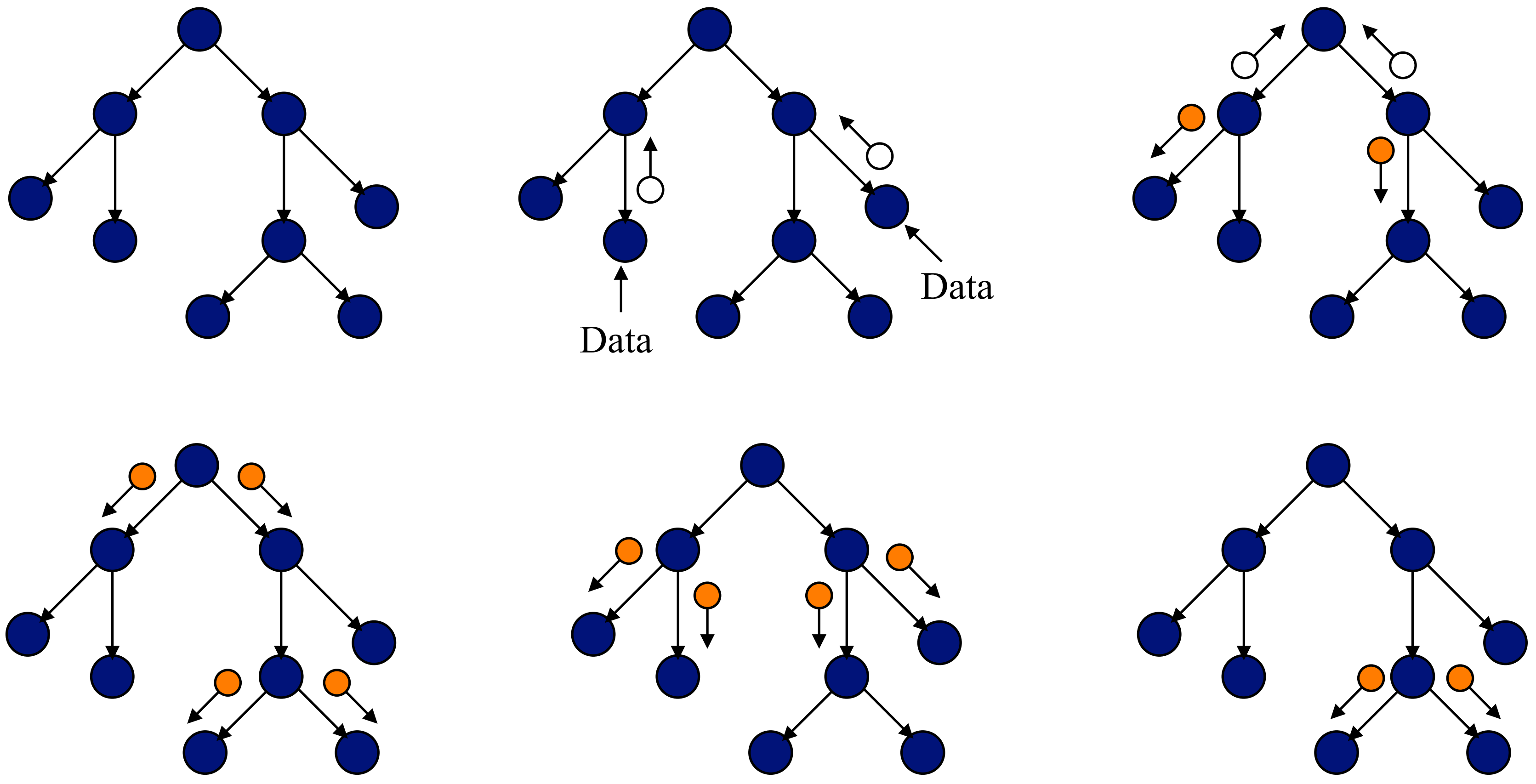
- The general probabilistic inference problem is to find the probability of an event given a set of evidence;

- This can be done in Bayesian nets with sequential applications of Bayes Theorem;

- In 1986 Judea Pearl published an innovative algorithm for performing inference in Bayesian nets.

Propagation Example

“The impact of each new piece of evidence is viewed as a perturbation that propagates through the network via message-passing between neighboring variables . . .” (Pearl, 1988, p 143`



- **The example above requires five time periods to reach equilibrium after the introduction of data**

Basic Inference

$$\begin{tikzcd} A \arrow[r] & B \end{tikzcd}$$

$$P(b) = ?$$

Product Rule

$$\begin{tikzcd} S \arrow[r] & C \end{tikzcd}$$

- $P(C,S) = P(C|S)\,P(S)$

$S\Downarrow \quad C\Rightarrow$	*none*	*benign*	*malignant*
no	0.768	0.024	0.008
light	0.132	0.012	0.006
heavy	0.035	0.010	0.005

Marginalization

$S\Downarrow \quad C\Rightarrow$	*none*	*benign*	*malig*	total
no	0.768	0.024	0.008	.80
light	0.132	0.012	0.006	.15
heavy	0.035	0.010	0.005	.05
total	0.935	0.046	0.019	

$\left.\begin{matrix}.80\\.15\\.05\end{matrix}\right\} P(Smoke)$

$\underbrace{0.935 \quad 0.046 \quad 0.019}_{P(Cancer)}$

Basic Inference

$$\begin{tikzcd} \boxed{A} \arrow[r] & \boxed{B} \arrow[r] & \boxed{C} \end{tikzcd}$$

$$\underbrace{P(b)} = \sum_a P(a, b) = \sum_a P(b \mid a)\, P(a)$$

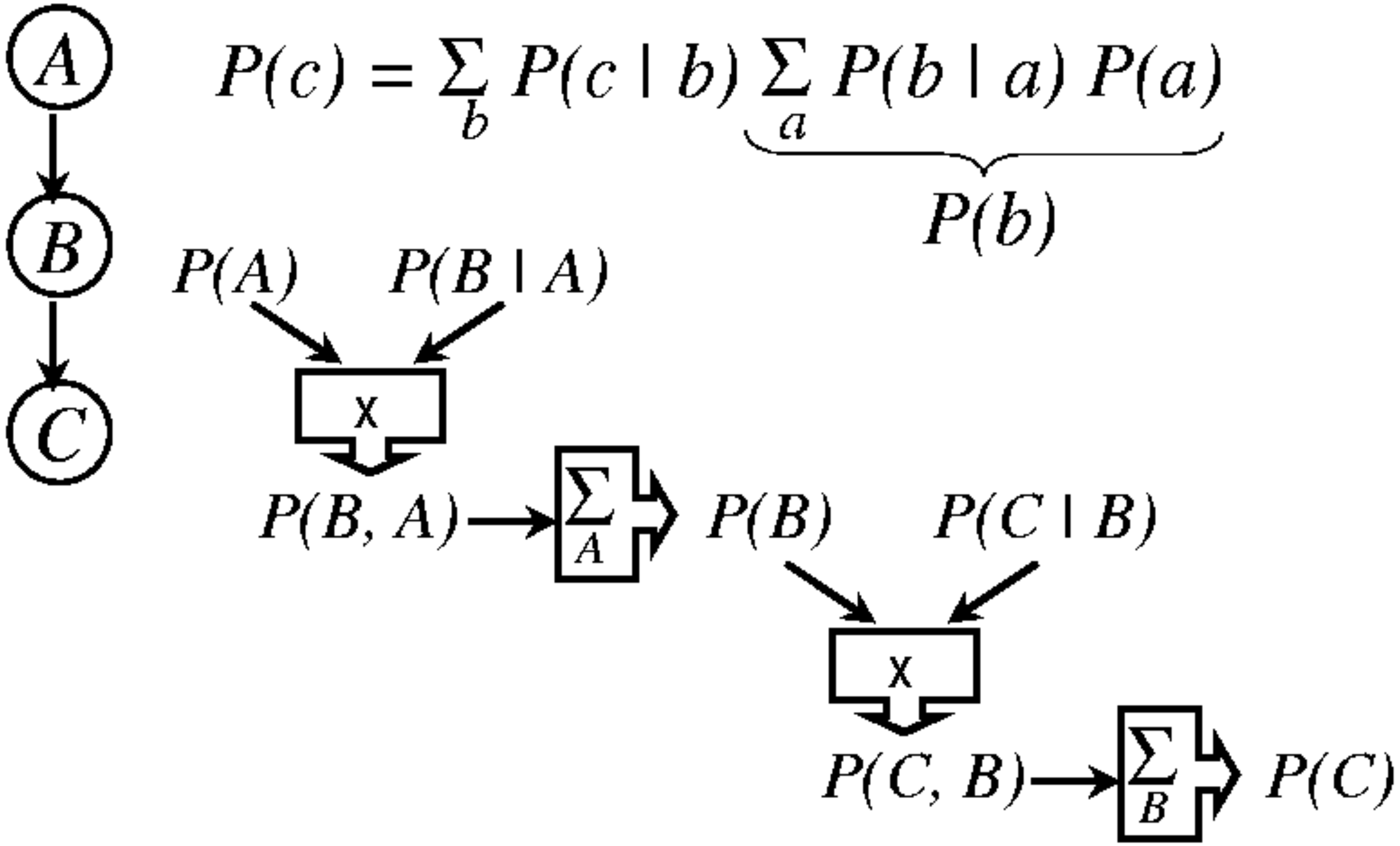
$$P(c) = \sum_b P(c \mid b)\, \overbrace{P(b)}$$

$$P(c) = \sum_{b,a} P(a, b, c) = \sum_{b,a} P(c \mid b)\, P(b \mid a)\, P(a)$$

$$= \sum_b P(c \mid b) \underbrace{\sum_a P(b \mid a)\, P(a)}_{P(b)}$$

Variable elimination

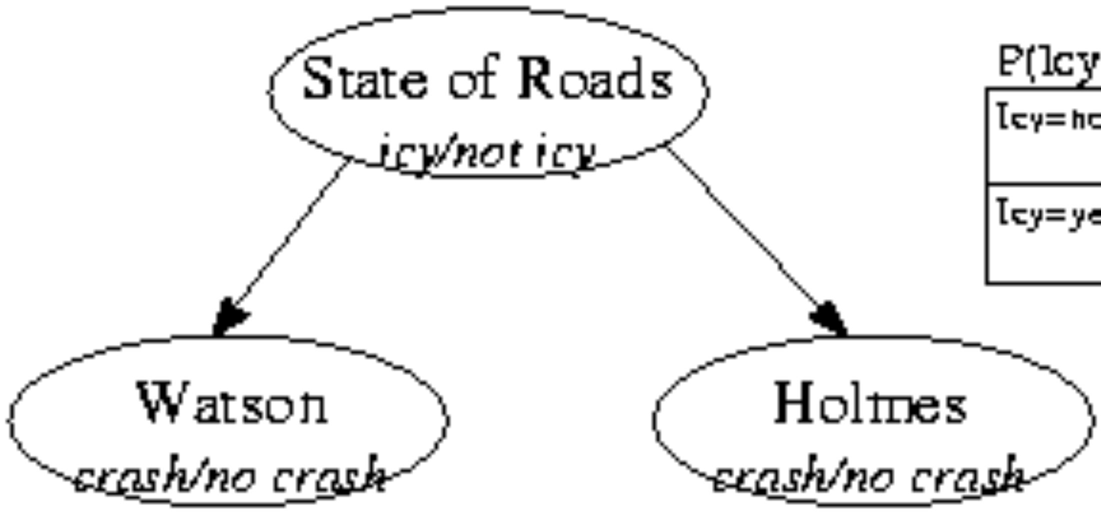
$$P(c) = \sum_b P(c \mid b) \underbrace{\sum_a P(b \mid a)\, P(a)}_{P(b)}$$



"Icy roads" example

- Inspector Smith is waiting for Holmes and Watson who are both late for an appointment.
- Smith is worried that if the roads are icy one or both of them may have crashed his car.
- Suddenly Smith learns that Watson has crashed.
- Smith thinks: *If Watson has crashed, probably the roads are icy, then Holmes has probably crashed too!*
- Smith then learns it is warm outside and roads are salted
- Smith thinks: *Watson was unlucky; Holmes should still make it.*

Bayes net for “Icy roads” example



P(Icy)

Icy=no	0.3
Icy=yes	0.7

P(Watson I Icy)	Icy = yes	Icy = no
Watson Crash = yes	0.8	0.1
Watson Crash = no	0.2	0.9

P(Holmes I Icy)	Icy = yes	Icy = no
Holmes Crash = yes	0.8	0.1
Holmes Crash = no	0.2	0.9

Extracting marginals

To find P(Holmes Crash) we first compute P(Holmes Crash, Icy) using the fundamental rule:

e.g. P(H Crash = yes, Icy = yes)

= P(H Crash=yes | Icy =yes)P(Icy=yes)

P(Holmes,Icy)	Icy = yes	Icy = no	P(H Crash)
Holmes Crash = yes	0.8 x0,7=0,56	0.1 x0,3=0,03	0.56+0.03=0.59
Holmes Crash = no	0.2 x0,7=0,14	0.9 x0,3=0,27	0.14+0.27=0.41

Then summing each row gives us the requred probabilities.
By symmetry P (W Crash) is the same.

Updating with Bayes rule (given evidence “Watson has crashed”)

After we discover that Watson has crashed we can compute P(Icy | W Crash =y) using Bayes rule:

$$P(\text{Icy} \mid \text{W Crash}=y) = \frac{P(\text{W Crash} = y \mid \text{Icy})P(\text{Icy})}{P(\text{W Crash} = y)}$$

$$= (0.8\text{x}0.7, 0.1\text{x}0.3)/0.59$$

$$= (0.95, 0.05)$$

Extracting the marginal

- To calculate P(H Crash | W Crash = y) we first calculate P(H Crash, Icy | W Crash)

P(H | W=y, Icy)	Icy = yes	Icy = no	
Holmes Crash = yes	0.8 x0.95=0,76	0.1 x0,05=0,005	0.765
Holmes Crash = no	0.2 x0.95=0,19	0.9 x0,05=0,045	0.235

Again, summing gives us P(H Crash | W Crash = yes)

P(H Crash | W Crash, Icy=no) = P(H Crash | Icy=no)
= (0.1,0.9)

Alternative perspective

Icy = no	0.3
Icy = yes	0.7

	Icy = yes	Icy = no
Watson Crash = yes	0.56	0.03
Watson Crash = no	0.14	0.27

	Icy = yes	Icy = no
Holmes Crash = yes	0.56	0.03
Holmes Crash = no	0.14	0.27

We represent the model as two joint tables, P(Watson, Icy) and P(Holmes, Icy) with a table for the overlap P(Icy).

Alternative perspective

Icy=no	0.3
Icy=yes	0.7

	Icy = yes	Icy = no
Watson Crash = yes	0.56/0.59 = 0.95	0.03/0.59= 0.05
Watson Crash = no	0	0

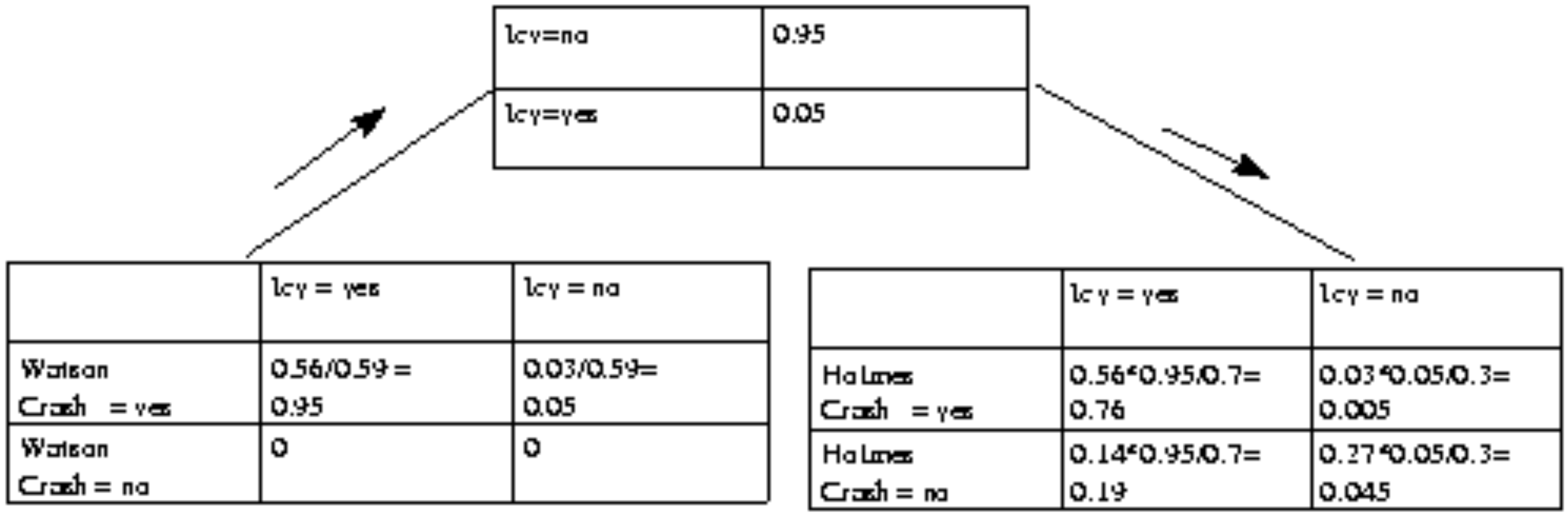
	Icy = yes	Icy = no
Holmes Crash = yes	0.56	0.03
Holmes Crash = no	0.14	0.27

If evidence on Watson arrives of the form $P^{New}(\text{W Crash}) = (1,0)$

then $P^{New}(\text{W Crash, Icy})$

$$= P(\text{Icy} \mid \text{W Crash})\, P^{New}(\text{W Crash}) = \frac{P(\text{W Crash, Icy})}{P(\text{W Crash})}\, P^{New}(\text{W Crash})$$

Alternative perspective



Icy=no	0.95
Icy=yes	0.05

	Icy = yes	Icy = no
Watson Crash = yes	0.56/0.59 = 0.95	0.03/0.59= 0.05
Watson Crash = no	0	0

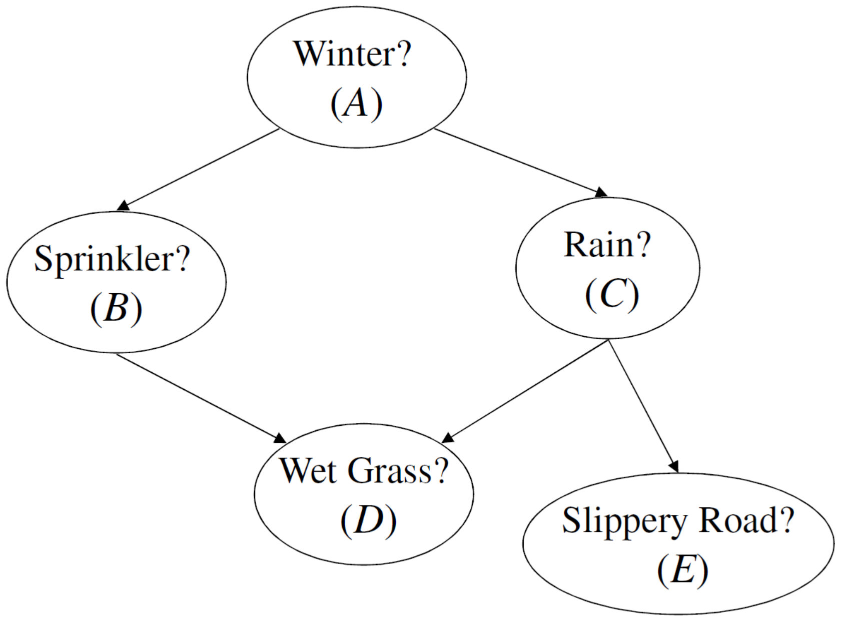
	Icy = yes	Icy = no
Holmes Crash = yes	0.56*0.95/0.7= 0.76	0.03*0.05/0.3= 0.005
Holmes Crash = no	0.14*0.95/0.7= 0.19	0.27*0.05/0.3= 0.045

The table for Icy can then be updated by marginalizing the table for Watson. The table for Holmes can then be updated using the same rule:

$$P^{New}(H\ Crash, Icy) = \frac{P(H\ Crash, Icy)}{P(Icy)} P^{New}(Icy)$$

Variable Elimination

- One of the simplest algorithms for inference in Bayesian networks
  - Successively remove variables from the Bayesian network until only the query variables remain



A	Θ_A
true	.6
false	.4

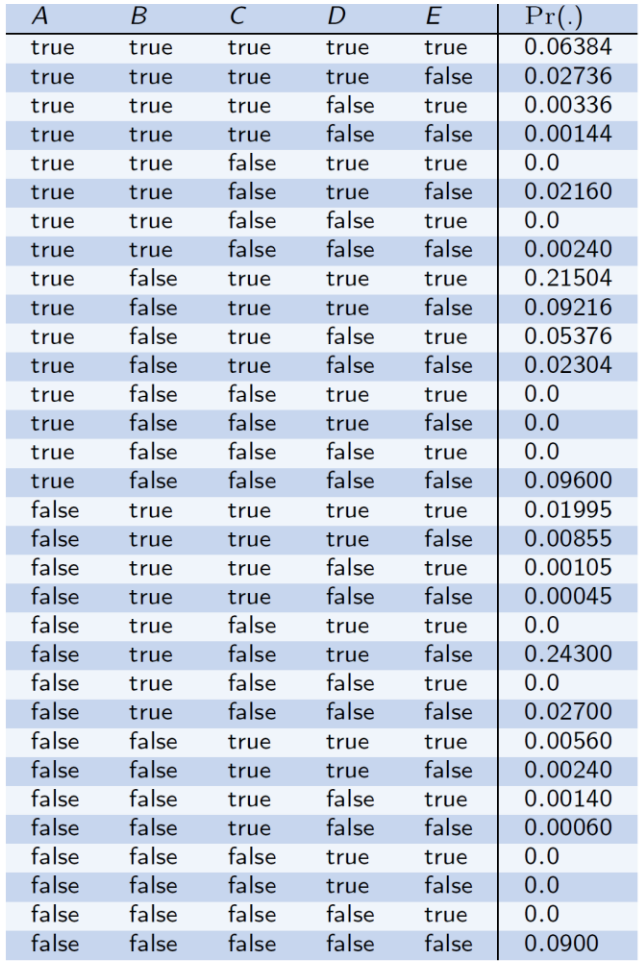
A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D\|BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Joint Probability Distribution



A	B	C	D	E	$\Pr(.)$
true	true	true	true	true	0.06384
true	true	true	true	false	0.02736
true	true	true	false	true	0.00336
true	true	true	false	false	0.00144
true	true	false	true	true	0.0
true	true	false	true	false	0.02160
true	true	false	false	true	0.0
true	true	false	false	false	0.00240
true	false	true	true	true	0.21504
true	false	true	true	false	0.09216
true	false	true	false	true	0.05376
true	false	true	false	false	0.02304
true	false	false	true	true	0.0
true	false	false	true	false	0.0
true	false	false	false	true	0.0
true	false	false	false	false	0.09600
false	true	true	true	true	0.01995
false	true	true	true	false	0.00855
false	true	true	false	true	0.00105
false	true	true	false	false	0.00045
false	true	false	true	true	0.0
false	true	false	true	false	0.24300
false	true	false	false	true	0.0
false	true	false	false	false	0.02700
false	false	true	true	true	0.00560
false	false	true	true	false	0.00240
false	false	true	false	true	0.00140
false	false	true	false	false	0.00060
false	false	false	true	true	0.0
false	false	false	true	false	0.0
false	false	false	false	true	0.0
false	false	false	false	false	0.0900

P(D=true,E=true)=?

P(A=true|D=true,E=true)=?

How does the algorithm work? Task: Computing probability of evidence

- Instantiate Evidence variables and remove them from all conditional probability tables
- Select an ordering of variables
- Eliminate variables one by one along the ordering
- How to eliminate a variable ?
  - **Multiply** all functions/factors that mention the variable yielding a function f
  - **Sum-out** the variable from f yielding a function f’
  - Add f’ to the set of original functions

Multiplication of factors

A	B	C	ϕ(A,B,C)
0	0	0	3
0	0	1	2
0	1	0	1
0	1	1	5
1	0	0	3
1	0	1	8
1	1	0	6
1	1	1	3

$\times$

A	C	D	ϕ(A,C,D)
0	0	0	4
0	0	1	2
0	1	0	11
0	1	1	4
1	0	0	2
1	0	1	1
1	1	0	5
1	1	1	1

$=$

A	B	C	D	ϕ(A,B,C,D)
0	0	0	0	3*4=12
0	0	0	1	3*2=6
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Complexity is the size of the product table (exp (w)) times the number of factors (m) where w is the cardinality of the union of the scopes of functions

Summing out a set of variables

A	B	C	ϕ(A,B,C)
0	0	0	3
0	0	1	2
0	1	0	1
0	1	1	5
1	0	0	3
1	0	1	8
1	1	0	6
1	1	1	3

Sum-out B and C

$\Longrightarrow$

A	ϕ(A)
0	
1	

$$\sum_{b,c} \phi(a, b, c)$$

Complexity is the size of the table : exp (w)

The Formal Algorithm

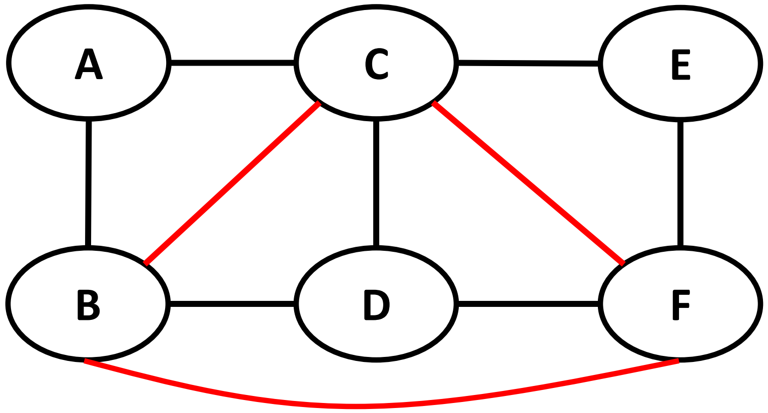
input:

$\mathcal{N}$:	Bayesian network
$\mathbf{Q}$:	variables in network $\mathcal{N}$
π:	ordering of network variables not in $\mathbf{Q}$

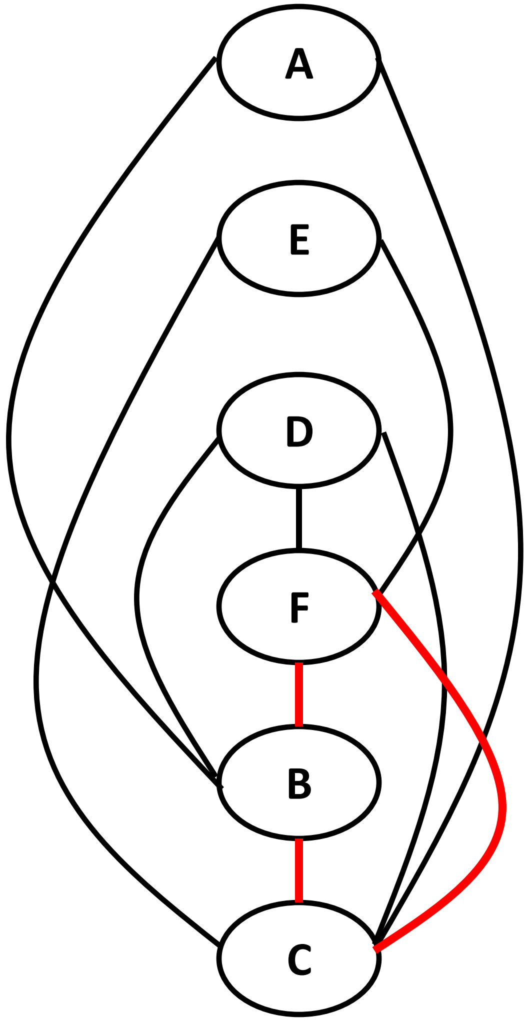
```
1: $\mathcal{S} \leftarrow$ CPTs of network $\mathcal{N}$
2: **for** $i = 1$ to length of order $\pi$ **do**
3:     $f \leftarrow \prod_k f_k$, where $f_k$ belongs to $\mathcal{S}$ and mentions variable $\pi(i)$
4:     $f_i \leftarrow \sum_{\pi(i)} f$
5:     replace all factors $f_k$ in $\mathcal{S}$ by factor $f_i$
6: **end for**
7: **return** $\prod_{f \in \mathcal{S}} f$
```

Variable Elimination: Complexity

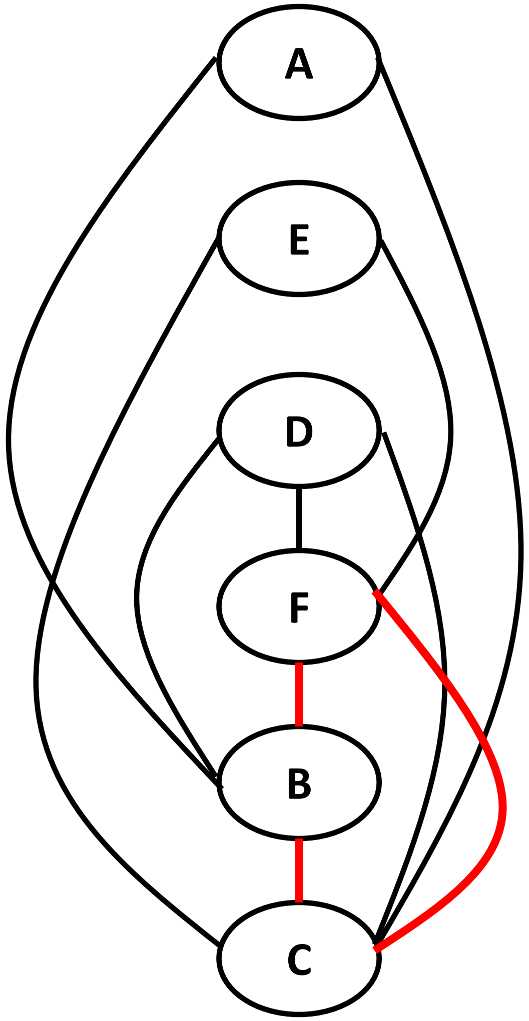
- Schematic operation on a graph



  - Process nodes in order
  - Eliminate = Connect all children of a node to each other



Variable elimination: Complexity



- Complexity of eliminating variable "i"
  - exp(children$_i$)
- Complexity of variable elimination:
  - nexp(max(children$_i$))
- Treewidth
  - Minimum over all possible graphs constructed this way

Variable Elimination for MPE and MAR

- MARGINAL TASK
  - Ratio of two evidence probabilities
  - P(A=a|B=b)=P(A=a,B=b)/P(B=b)
  - Use VE to compute numerator and denominator
- MPE TASK
  - Replace sum-out operation by max-out operation

MAX_S

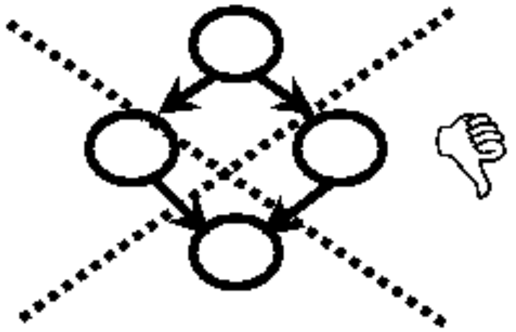
S	C	Value
male	yes	0.05
male	no	0.95
female	yes	0.01
female	no	0.99

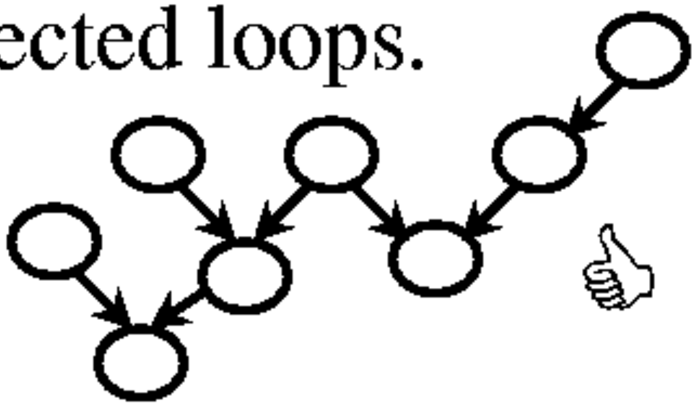
=

C	Value
yes	0.05
no	0.99

Polytrees

- A network is *singly connected* (a *polytree*) if it contains no undirected loops.

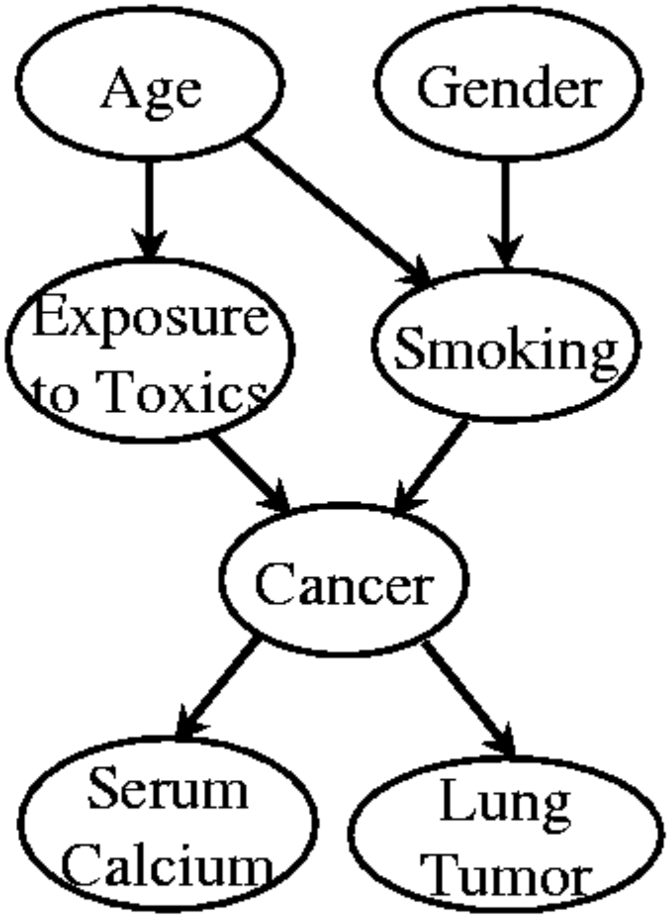


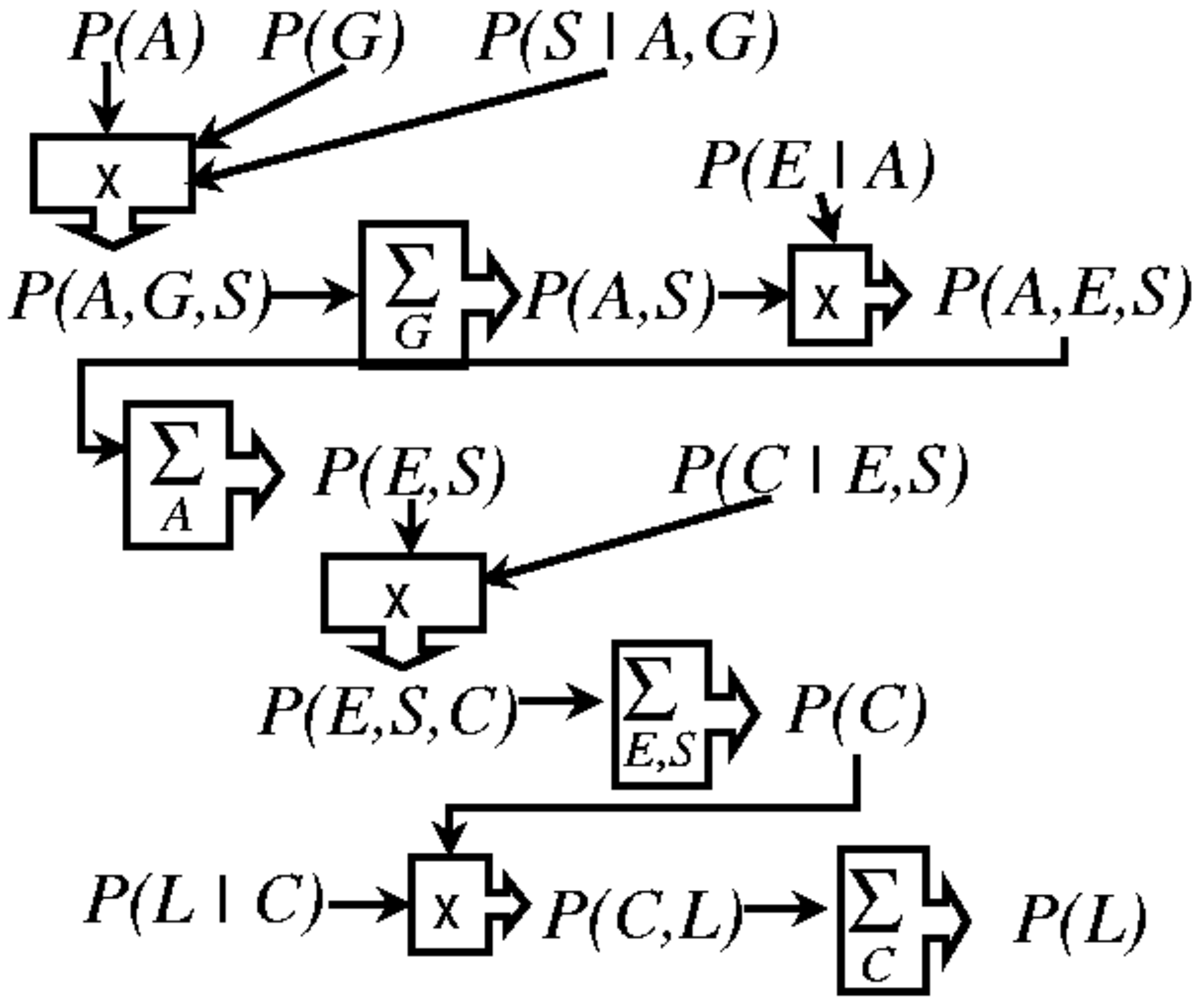


Theorem: Inference in a singly connected network can be done in linear time*.

Main idea: in variable elimination, need only maintain distributions over single nodes.

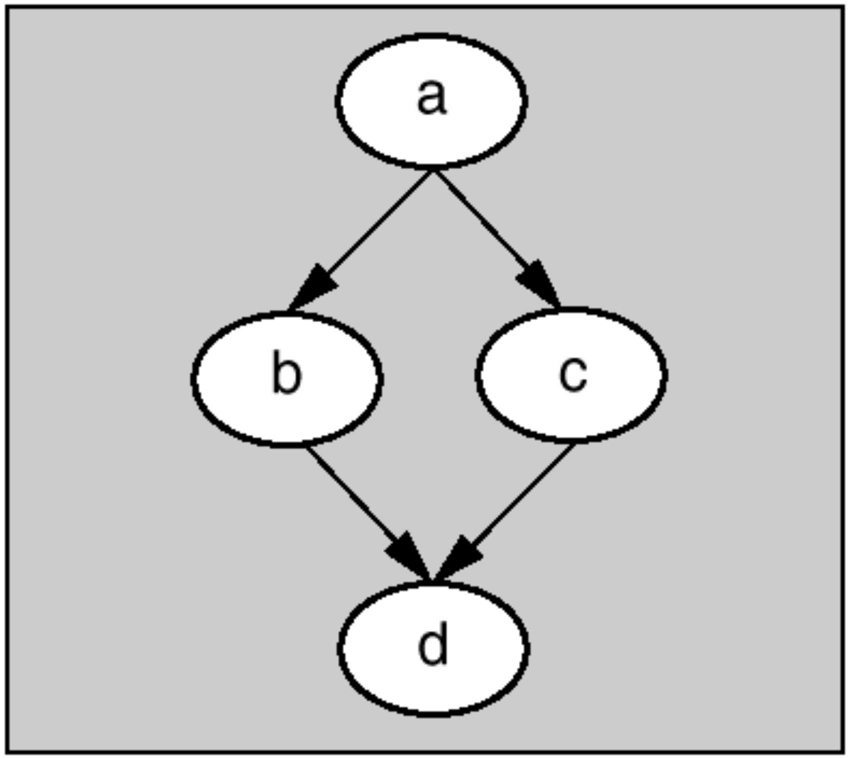
Variable Elimination with loops





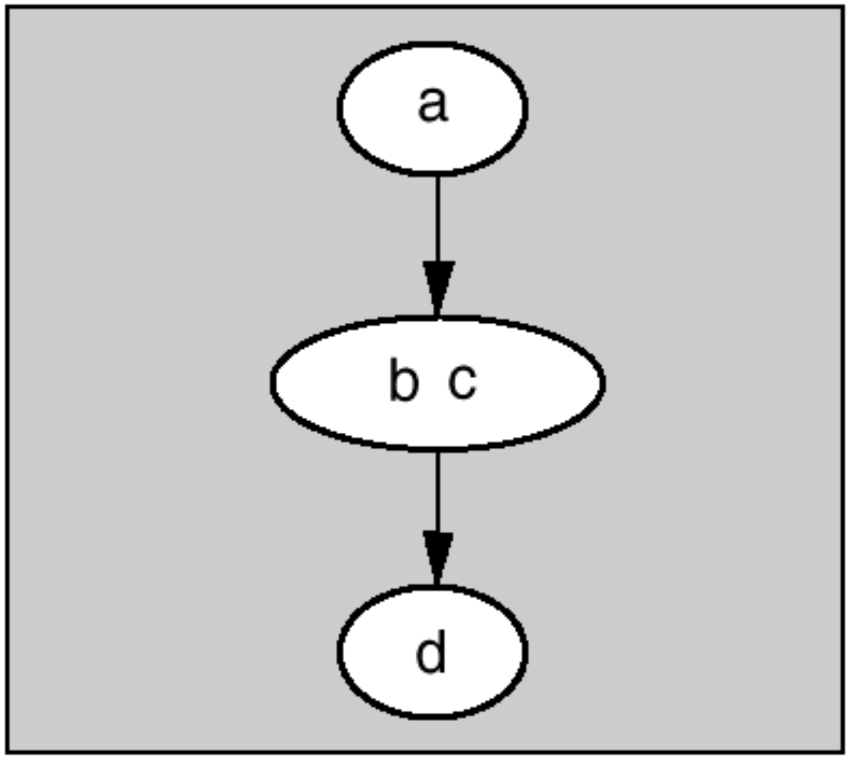
Complexity is exponential in the size of the factors

Join Trees



A Multiply Connected Network.

There are two paths between node a *and node* d.



A Clustered, Multiply Connected Network.

By clustering nodes b *and* c, *we turned the graph into a singly connected network.*

From Variable Elimination to the junction tree algorithms

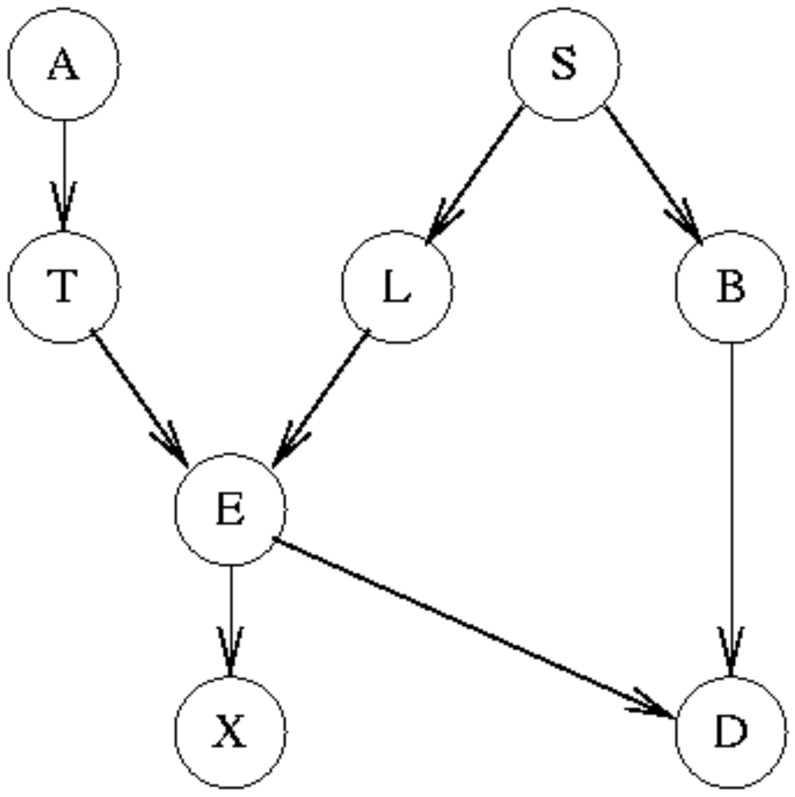
- Variable Elimination is **query sensitive**: we must specify the query variables in advance. This means each time we run a different query, we must re-run the entire algorithm.
- The **junction tree algorithms** generalize Variable Elimination to avoid this; they **compile** the density into a data structure that supports the **simultaneous** execution of a large class of queries.

Graphical Method of Building the Junction Tree

The Junction Tree can be constructed through a series of graph operations

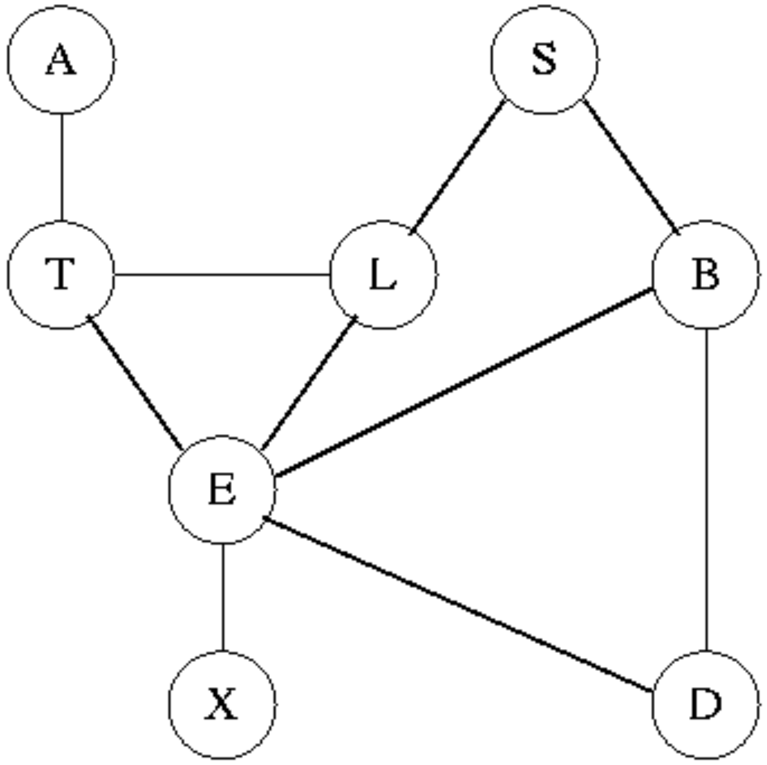
- **Marry the Parents** ("moralize the graph"): Add an undirected edge between every pair of parents of a node (unless they are already connected.
- **Make All Arrows Undirected**
- **Triangulate the Graph:** Add edges so that every cycle of length 4 or more contains a chord.
- **Identify the maximal Cliques:** A clique is a complete graph. A maximal clique is a maximal complete subgraph.
- **Form Junction Graph:** Create a cluster node for each clique and label it with the variables in the clique.
  Create an edge between any pair of cluster nodes that share variables.
  Place a separator node on the edge labeled with the set of variables shared by the cluster nodes it joins.
- **Form the junction tree:** Compute a maximum weighted spanning tree of the junction graph where the weight on each edge is the number of variables in the separator of the edge.

Example



$$P(U) = P(A)P(S)P(T|A)P(L|S)P(B|S)P(E|L,T)P(D|B,E)P(X|E)$$

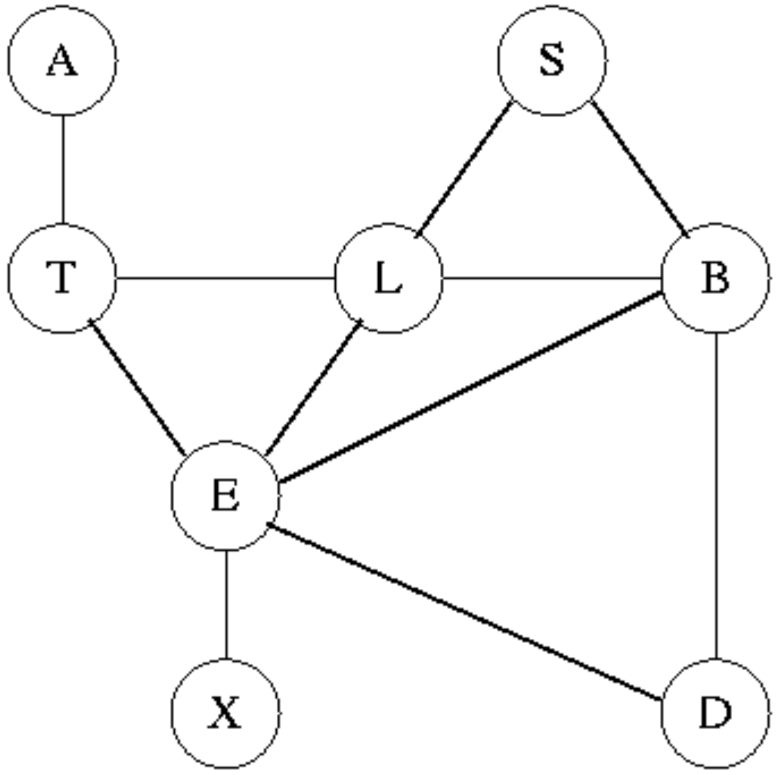
Step 1: Moralize the Graph



We join **T** and **L** because they are parents of **E**.

We join **E** and **B** because they are parents of **D**.

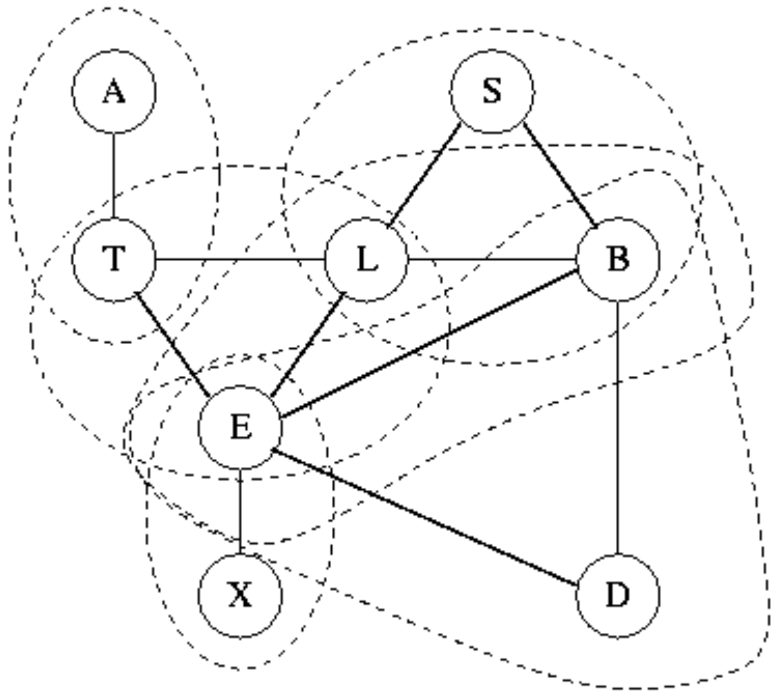
Step 2: Triangulate the Moral Graph

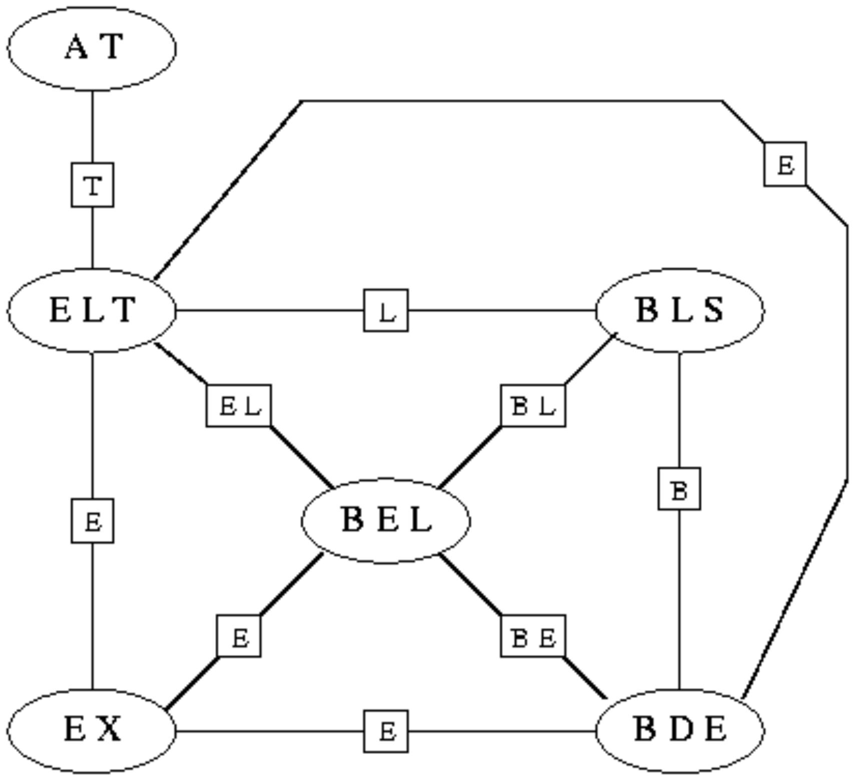


There is a cycle of length four with no shortcuts: **E**, **L**, **S**, **B**.

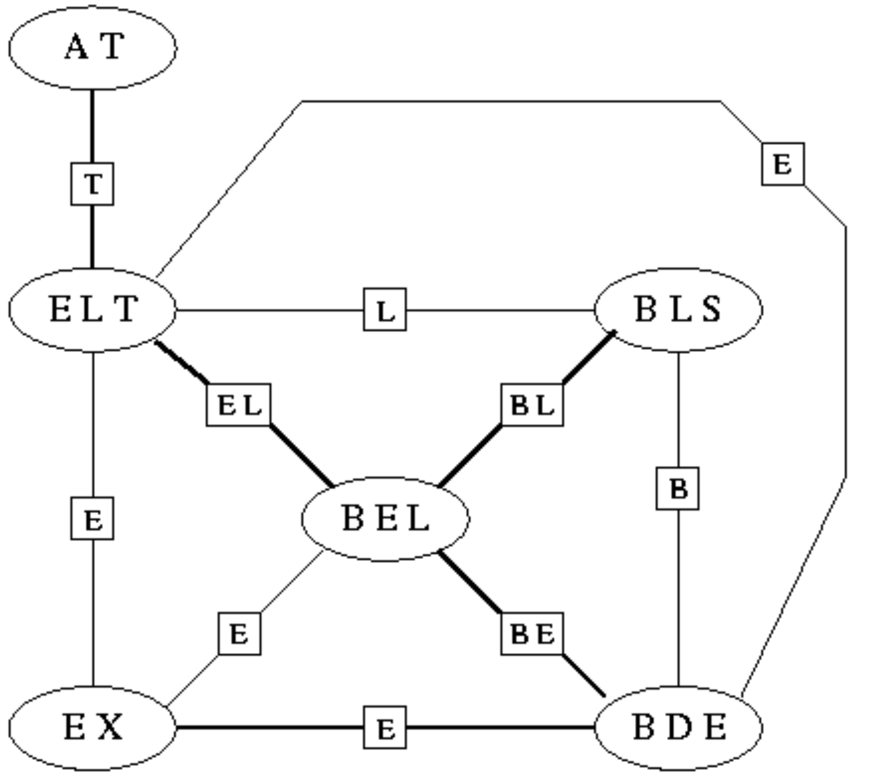
We have a choice of where to add the shortcut. Either **LB** or **SE** would work.

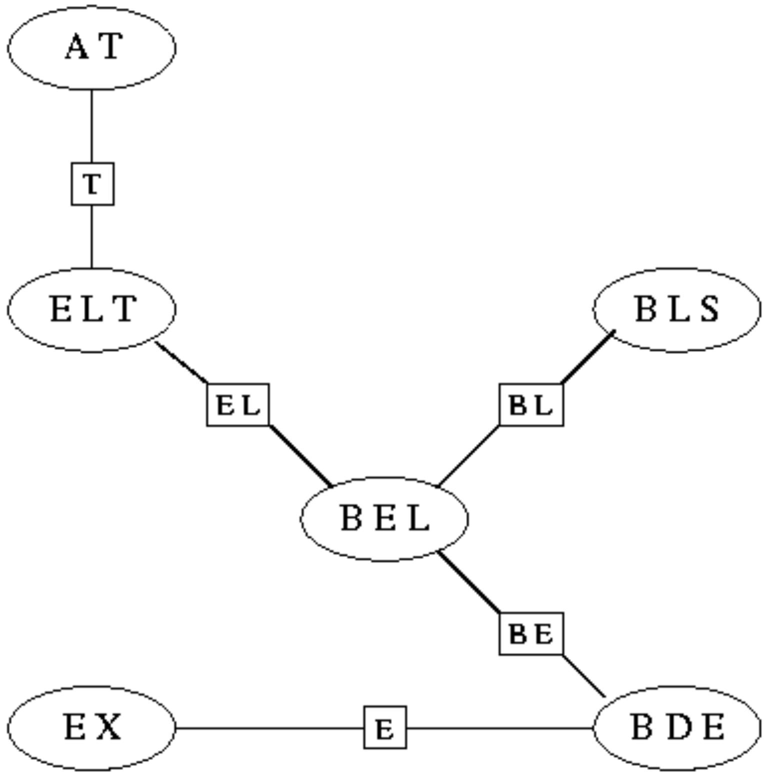
Step 3: Cliques and Junction Graph





Step 4: Junction Tree



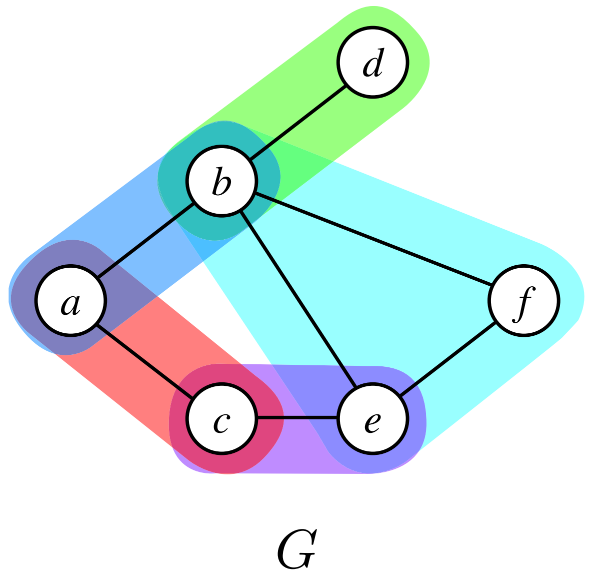


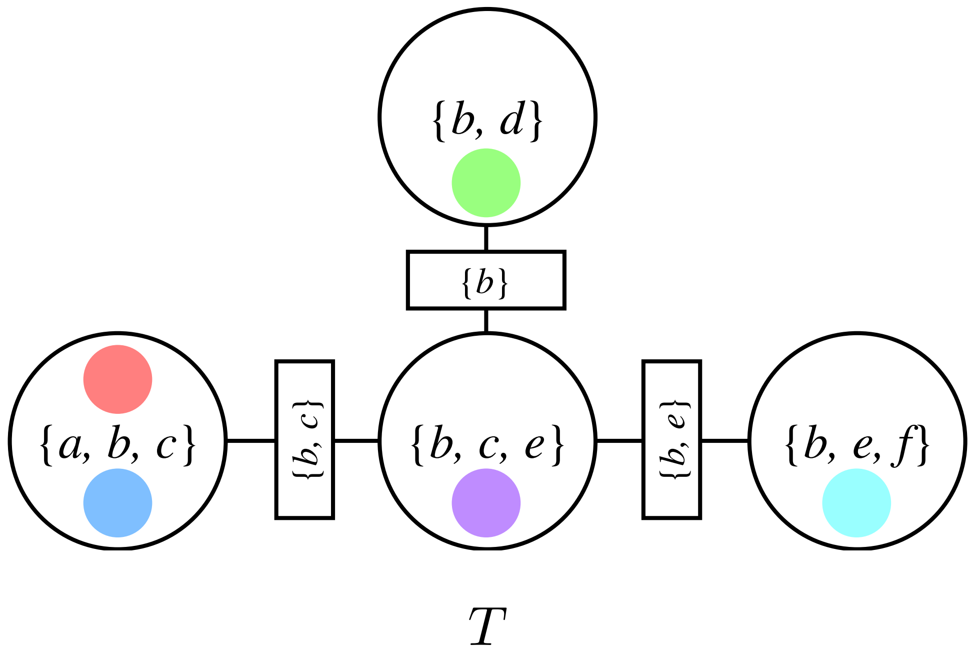
Notice that the running intersection property holds (this is guaranteed by the maximum weight spanning tree and the moralizing and triangulating edges).

The Junction Tree Algorithm

- *Exact* inference on general graphs.
- Works by turning the initial graph into a *junction tree* and then running a sum-product-like algorithm.
- *Intractable* on graphs with large cliques.

Junction trees



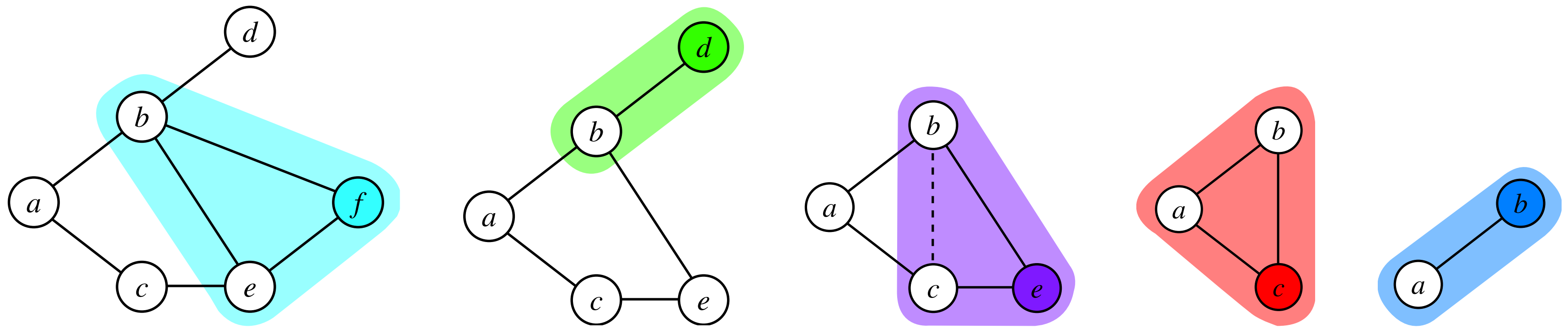


A cluster graph T is a **junction tree** for G if it has these three properties:

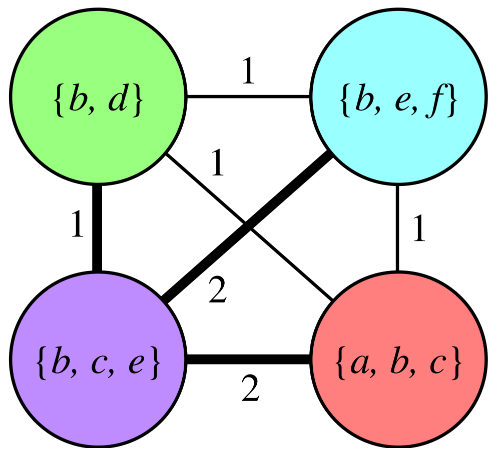
1. **singly connected**: there is exactly one path between each pair of clusters.
2. **covering**: for each clique A of G there is some cluster C such that $A \subseteq C$.
3. **running intersection**: for each pair of clusters B and C that contain i, each cluster on the unique path between B and C also contains i.

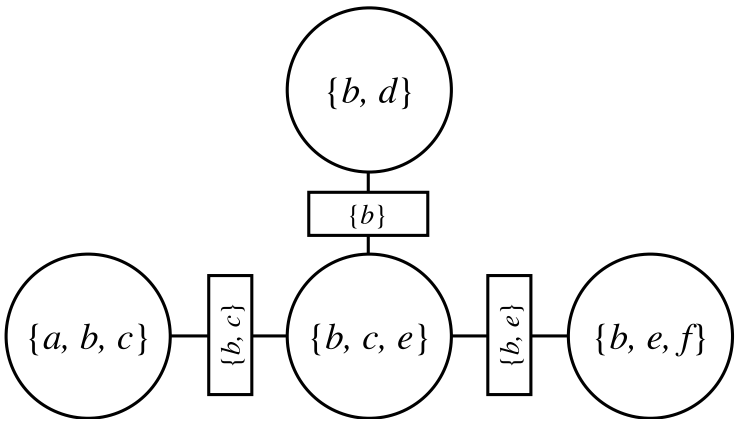
An example of building junction trees

1. Compute the elimination cliques (the order here is f, d, e, c, b, a).



2. Form the complete cluster graph over the maximal elimination cliques and find a maximum-weight spanning tree.





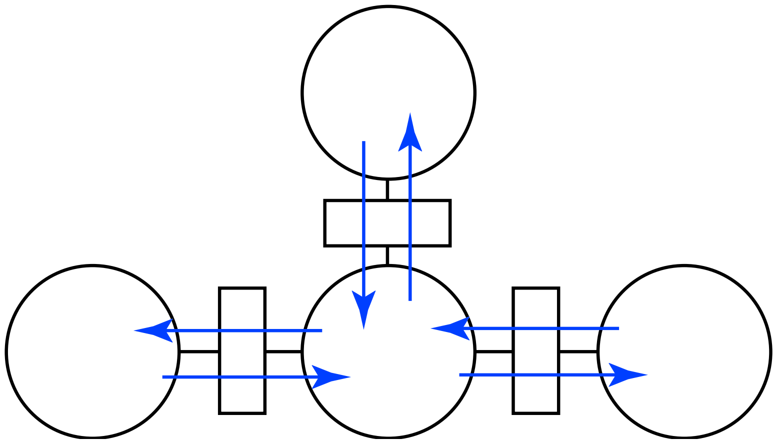
Decomposable densities

- A factorized density
$$p(\mathbf{u}) = \frac{1}{Z} \prod_{C \in \mathbf{C}} \psi_C(\mathbf{u}_C)$$
is **decomposable** if there is a junction tree with cluster set $\mathbf{C}$.
- To convert a factorized density p to a decomposable density:
  1. Build a junction tree T for the Markov graph of p.
  2. Create a potential ψ_C for each cluster C of T and initialize it to unity.
  3. Multiply each potential ψ of p into the cluster potential of one cluster that covers its variables.
- Note: this is possible only because of the **covering** property.

The junction tree inference algorithms

The junction tree algorithms take as input a decomposable density and its junction tree. They have the same distributed structure:

- Each cluster starts out knowing only its local potential and its neighbors.
- Each cluster sends one message (potential function) to each neighbor.
- By combining its local potential with the messages it receives, each cluster is able to compute the marginal density of its variables.



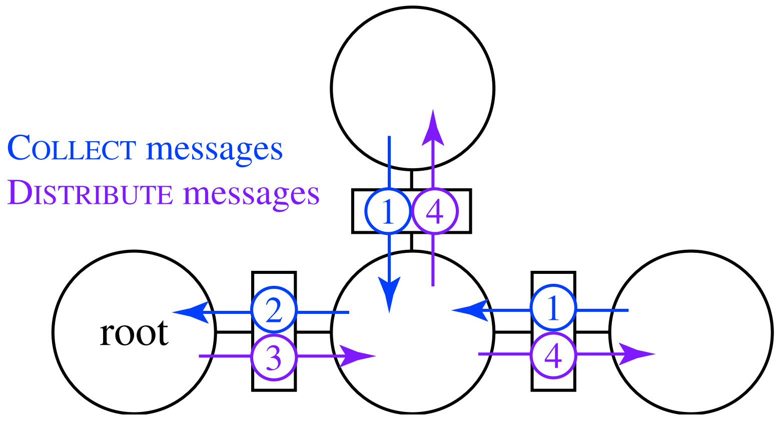
The message passing protocol

The junction tree algorithms obey the **message passing protocol**:

> Cluster B is allowed to send a message to a neighbor C only after it has received messages from all neighbors except C.

One admissible schedule is obtained by choosing one cluster R to be the root, so the junction tree is directed. Execute COLLECT(R) and then DISTRIBUTE(R):

1. COLLECT(C): For each child B of C, recursively call COLLECT(B) and then pass a message from B to C.
2. DISTRIBUTE(C): For each child B of C, pass a message to B and then recursively call DISTRIBUTE(B).



The Shafer–Shenoy Algorithm

- The **message sent from** B **to** C is defined as

$$\mu_{BC}(\mathbf{u}) \stackrel{\triangle}{=} \sum_{\mathbf{v} \in \mathbb{X}_{B \setminus C}} \psi_B(\mathbf{u} \cup \mathbf{v}) \prod_{\substack{(A,B) \in \mathbf{E} \\ A \neq C}} \mu_{AB}(\mathbf{u}_A \cup \mathbf{v}_A)$$

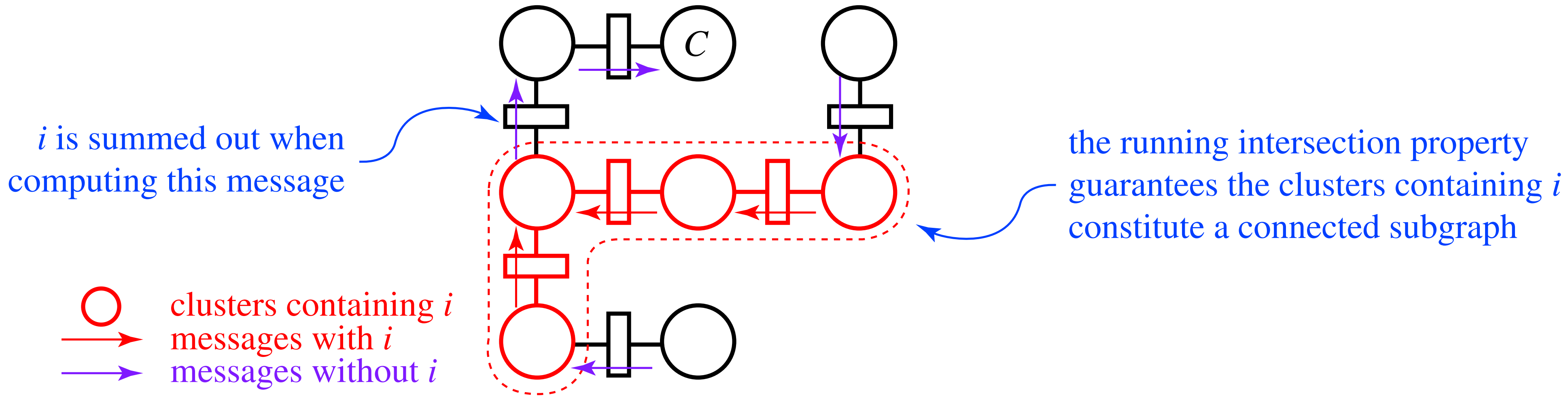
- Procedurally, cluster B computes the product of its local potential ψ_B and the messages from all clusters **except** C, marginalizes out all variables that are not in C, and then sends the result to C.
- Note: μ_{BC} is well-defined because the junction tree is **singly connected**.
- The **cluster belief at** C is defined as

$$\beta_C(\mathbf{u}) \stackrel{\triangle}{=} \psi_C(\mathbf{u}) \prod_{(B,C) \in \mathbf{E}} \mu_{BC}(\mathbf{u}_B)$$

This is the product of the cluster's local potential and the messages received from **all** of its neighbors. We will show that $\beta_C \propto p_C$.

Correctness: Shafer–Shenoy is Variable Elimination in all directions at once

- The cluster belief β_C is computed by alternatingly multiplying cluster potentials together and summing out variables.
- This computation is of the same basic form as Variable Elimination.
- To prove that $\beta_C \propto p_C$, we must prove that no sum is "pushed in too far".
- This follows directly from the **running intersection property**:



C
i is summed out when computing this message
the running intersection property guarantees the clusters containing i constitute a connected subgraph
clusters containing i
messages with i
messages without i

The HUGIN Algorithm

- Give each cluster C and each separator S a potential function over its variables. Initialize:

$$\begin{aligned} \phi_C(\mathbf{u}) &= \psi_C(\mathbf{u}) \\ \phi_S(\mathbf{u}) &= 1 \end{aligned}$$

- To pass a message from B to C over separator S, update

$$\begin{aligned} \phi_S^*(\mathbf{u}) &= \sum_{\mathbf{v} \in \mathbb{X}_{B \setminus S}} \phi_B(\mathbf{u} \cup \mathbf{v}) \\ \phi_C^*(\mathbf{u}) &= \phi_C(\mathbf{u}) \frac{\phi_S^*(\mathbf{u}_S)}{\phi_S(\mathbf{u}_S)} \end{aligned}$$

- After all messages have been passed, $\phi_C \propto p_C$ for all clusters C.

Correctness: HUGIN is a time-efficient version of Shafer–Shenoy

- Each time the Shafer–Shenoy algorithm sends a message or computes its cluster belief, it multiplies together messages.
- To avoid performing these multiplications repeatedly, the HUGIN algorithm caches in ϕ_C the running product of ψ_C and the messages received so far.
- When B sends a message to C, it **divides out** the message C sent to B from this running product.

Summary: the junction tree algorithms

Compile time:

1. Build the junction tree T:
   (a) Obtain a set of maximal elimination cliques with Node Elimination.
   (b) Build a weighted, complete cluster graph over these cliques.
   (c) Choose T to be a maximum-weight spanning tree.
2. Make the density decomposable with respect to T.

Run time:

1. Instantiate evidence in the potentials of the density.
2. Pass messages according to the message passing protocol.
3. Normalize the cluster beliefs/potentials to obtain conditional densities.

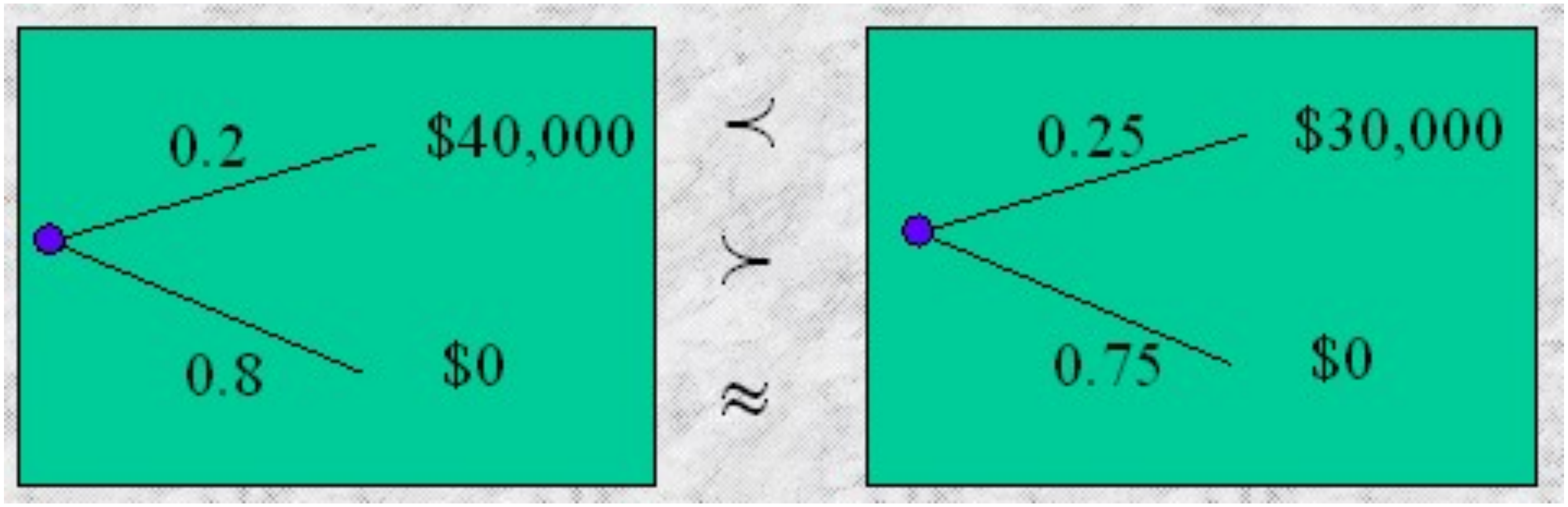
Summary

- The junction tree algorithms generalize Variable Elimination to the efficient, simultaneous execution of a large class of queries.
- The algorithms take the form of message passing on a graph called a junction tree, whose nodes are clusters, or sets, of variables.
- Each cluster starts with one potential of the factorized density. By combining this potential with the potentials it receives from its neighbors, it can compute the marginal over its variables.
- Two junction tree algorithms are the Shafer–Shenoy algorithm and the HUGIN algorithm, which avoids repeated multiplications.
- The complexity of the algorithms scales with the width of the junction tree.
- The algorithms can be generalized to solve other problems by using other commutative semirings.

Decision making

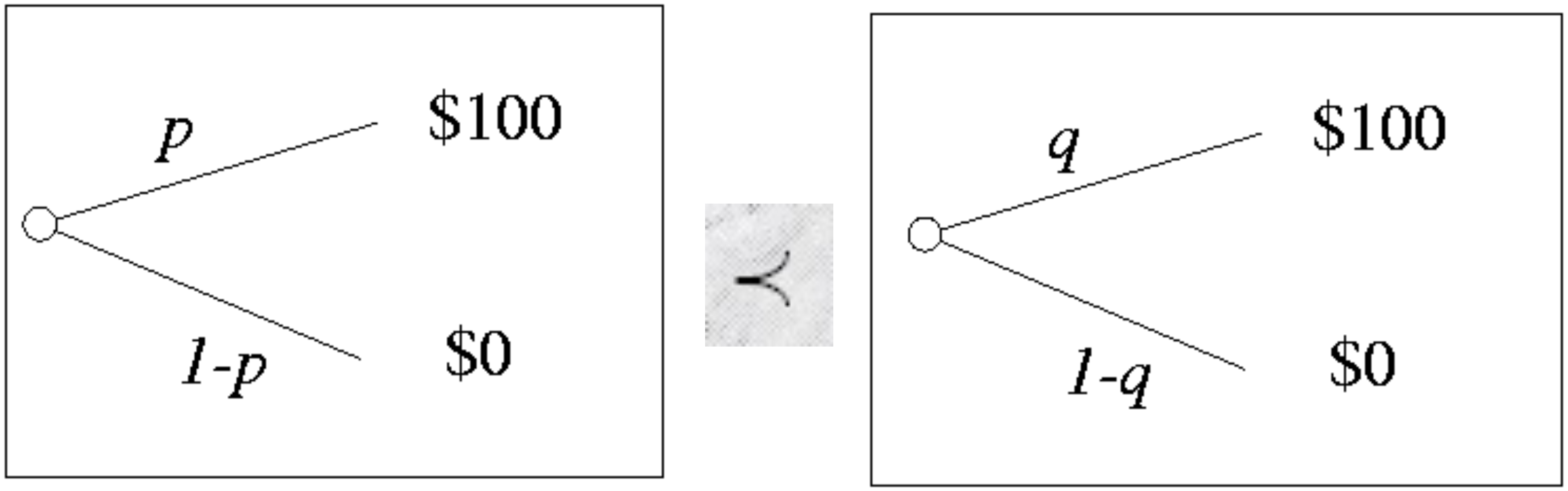
- Decision - an irrevocable allocation of domain resources
- Decision should be made so as to maximize expected utility.
- View decision making in terms of
  - Beliefs/Uncertainties
  - Alternatives/Decisions
  - Objectives/Utilities

Preference for Lotteries



Desired Properties for Preferences over Lotteries

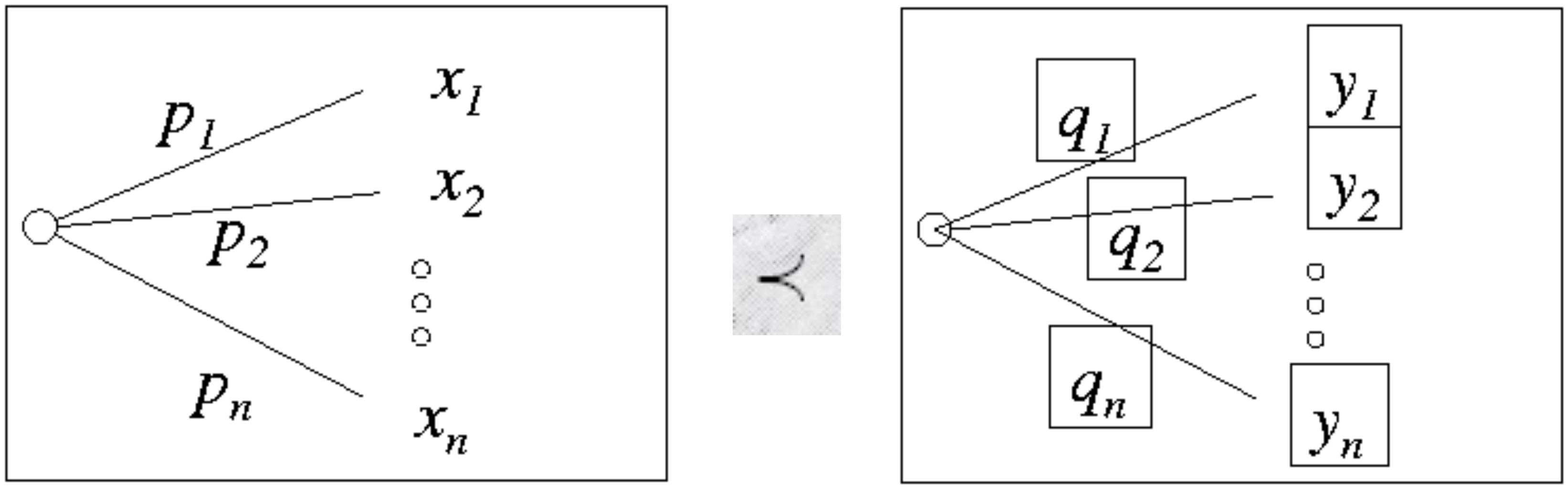
If you prefer \$100 to \$0 and $p < q$ then



(always)

Expected Utility

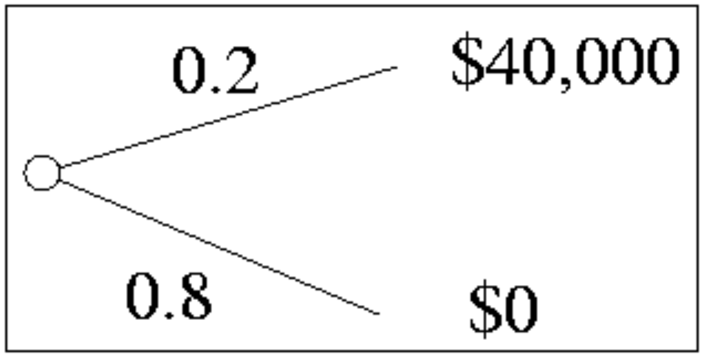
Properties of preference ⇒
existence of function *U*, that satisfies:



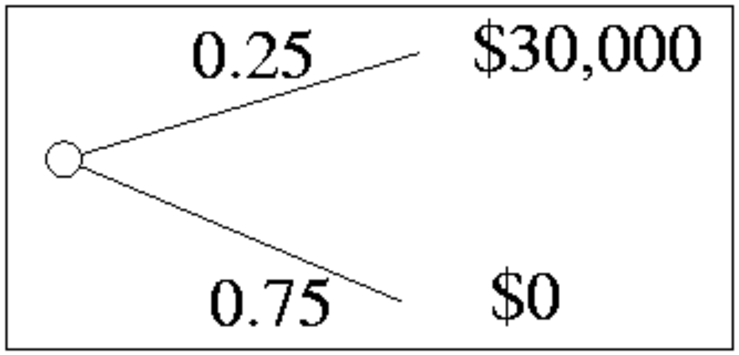
iff

$$\Sigma_i \, p_i \, U(x_i) \quad < \quad \Sigma_i \, q_i \, U(y_i)$$

Are people rational?

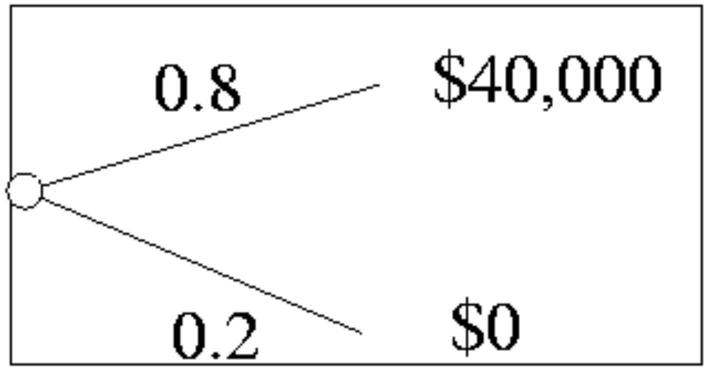


$\succ$

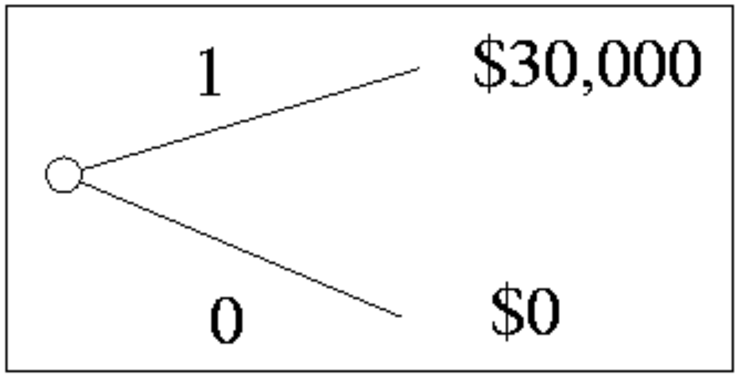


$$0.2 \cdot U(\$40k) > 0.25 \cdot U(\$30k)$$

$$0.8 \cdot U(\$40k) > U(\$30k)$$

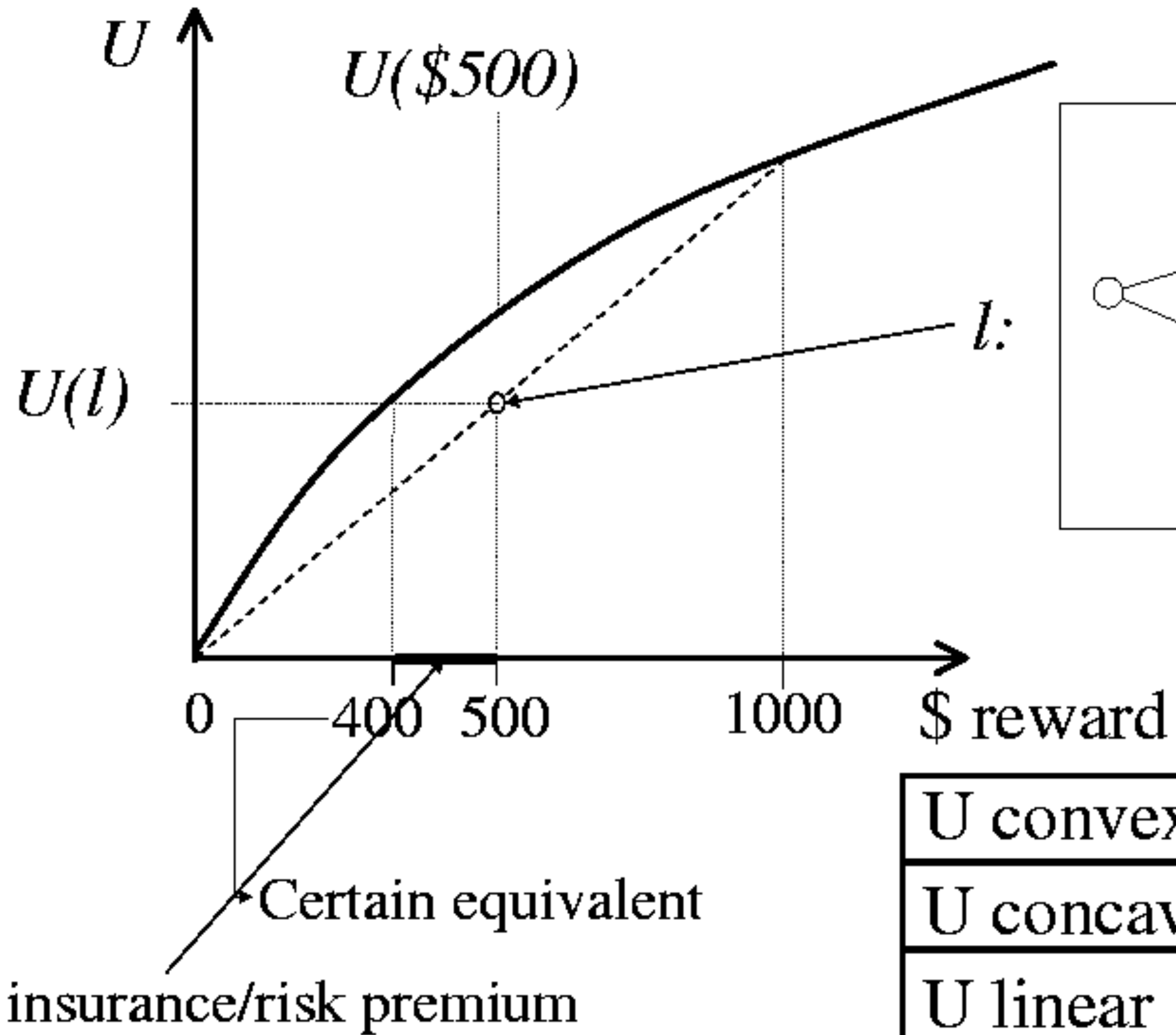


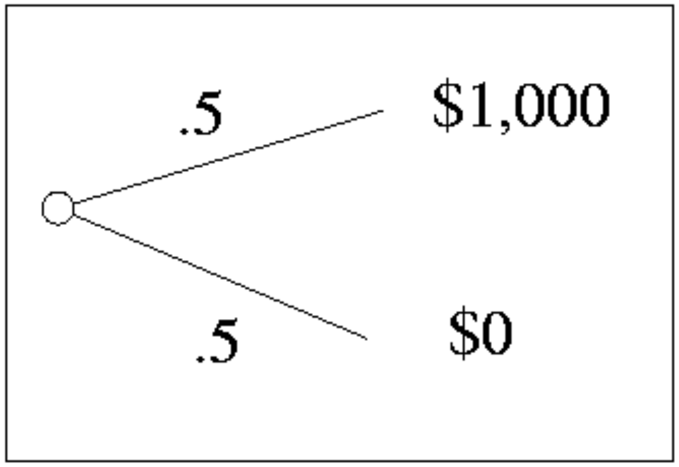
$\prec$



$$0.8 \cdot U(\$40k) < U(\$30k)$$

Attitudes towards risk

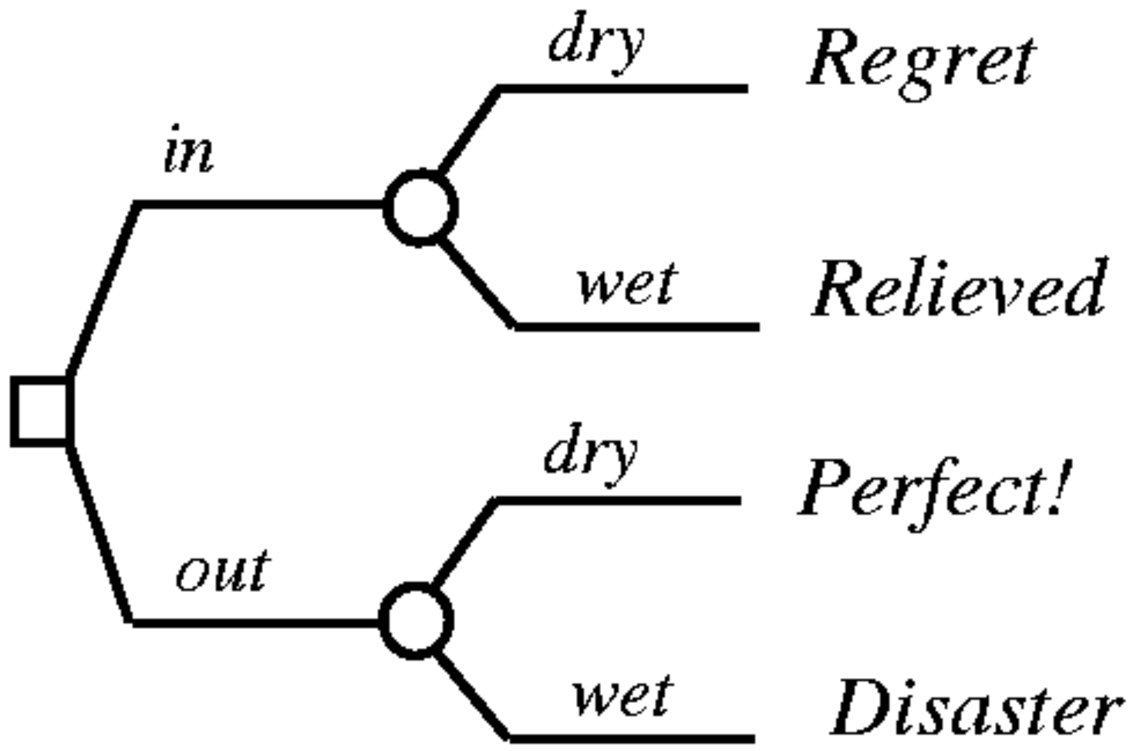




U convex	risk averse
U concave	risk seeking
U linear	risk neutral

A Decision Problem

Should I have my party inside or outside?

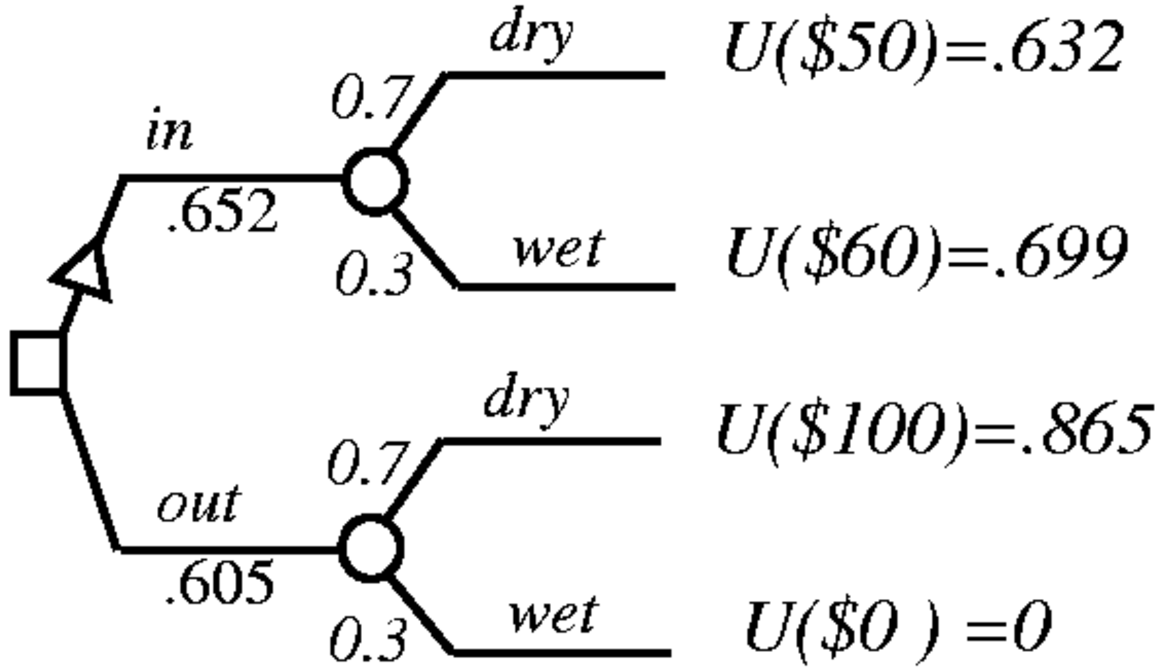


Value Function

- A numerical score over all possible states of the world.

Location?	Weather?	Value
in	dry	\$50
in	wet	\$60
out	dry	\$100
out	wet	\$0

Maximizing Expected Utility



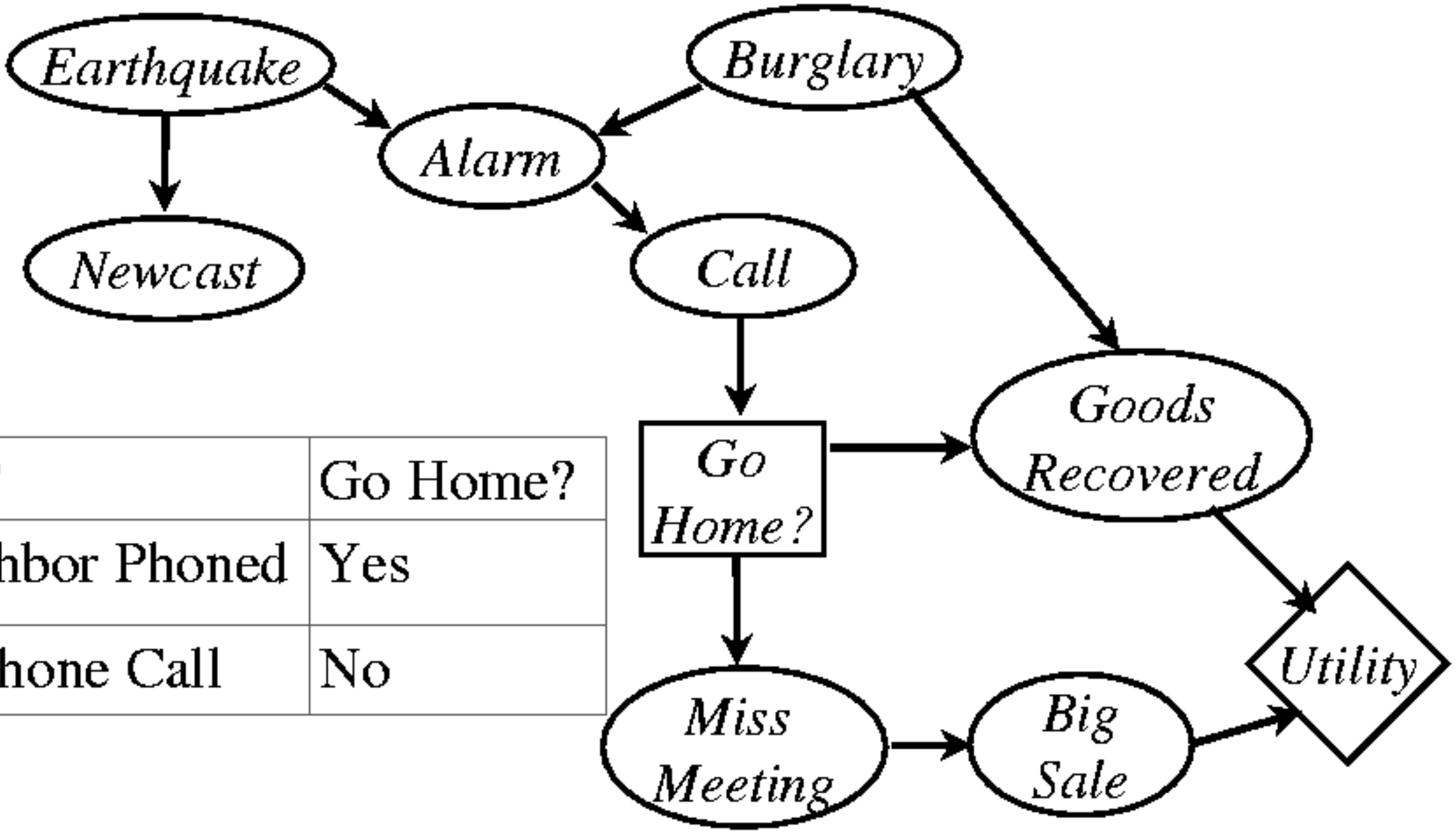
choose the action that maximizes expected utility

$EU(in) = 0.7 \cdot .632 + 0.3 \cdot .699 = .652$

$\Longrightarrow$ Choose *in*

$EU(out) = 0.7 \cdot .865 + 0.3 \cdot 0 = .605$

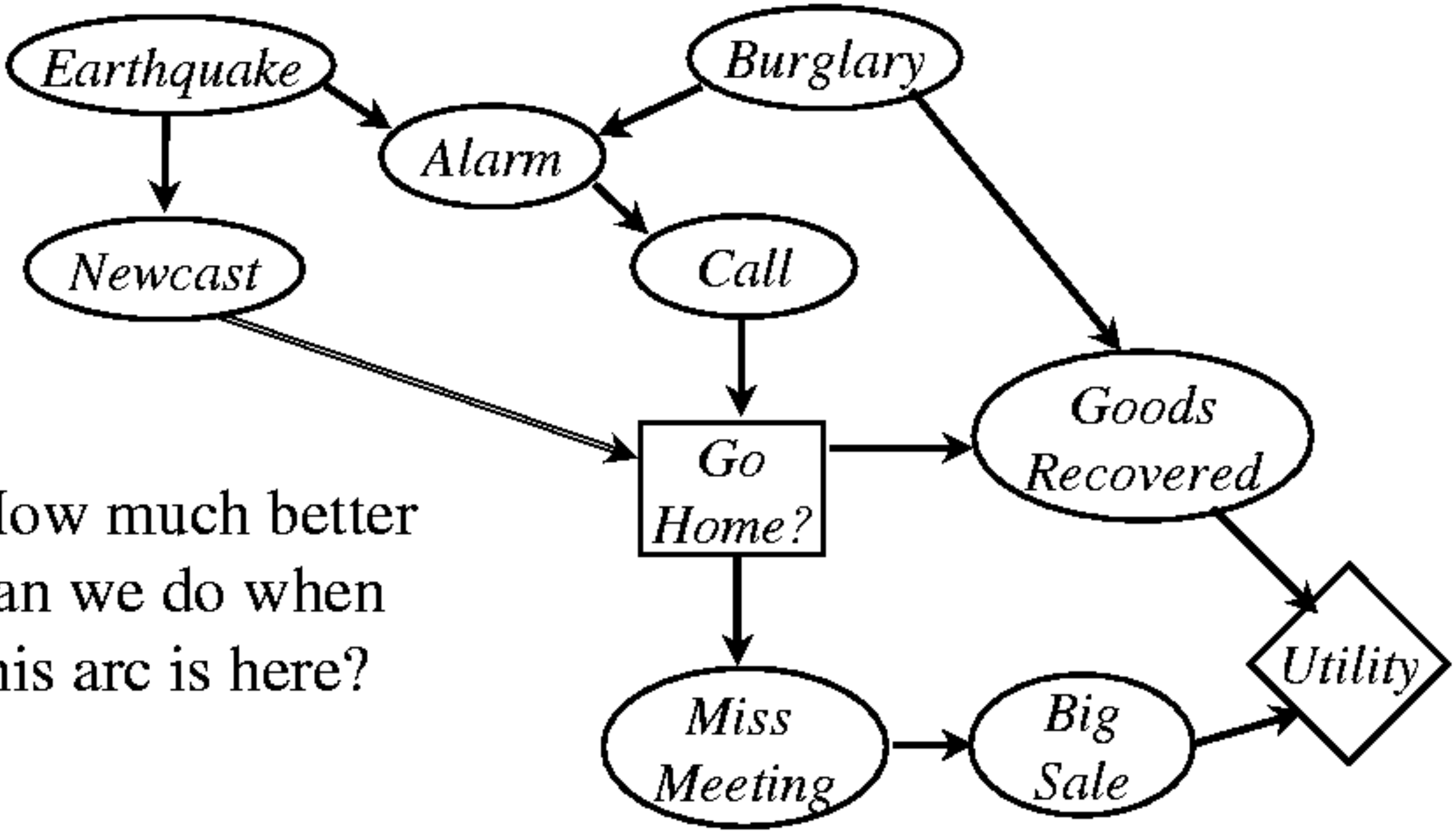
Decision Making with Influence Diagrams



Call?	Go Home?
Neighbor Phoned	Yes
No Phone Call	No

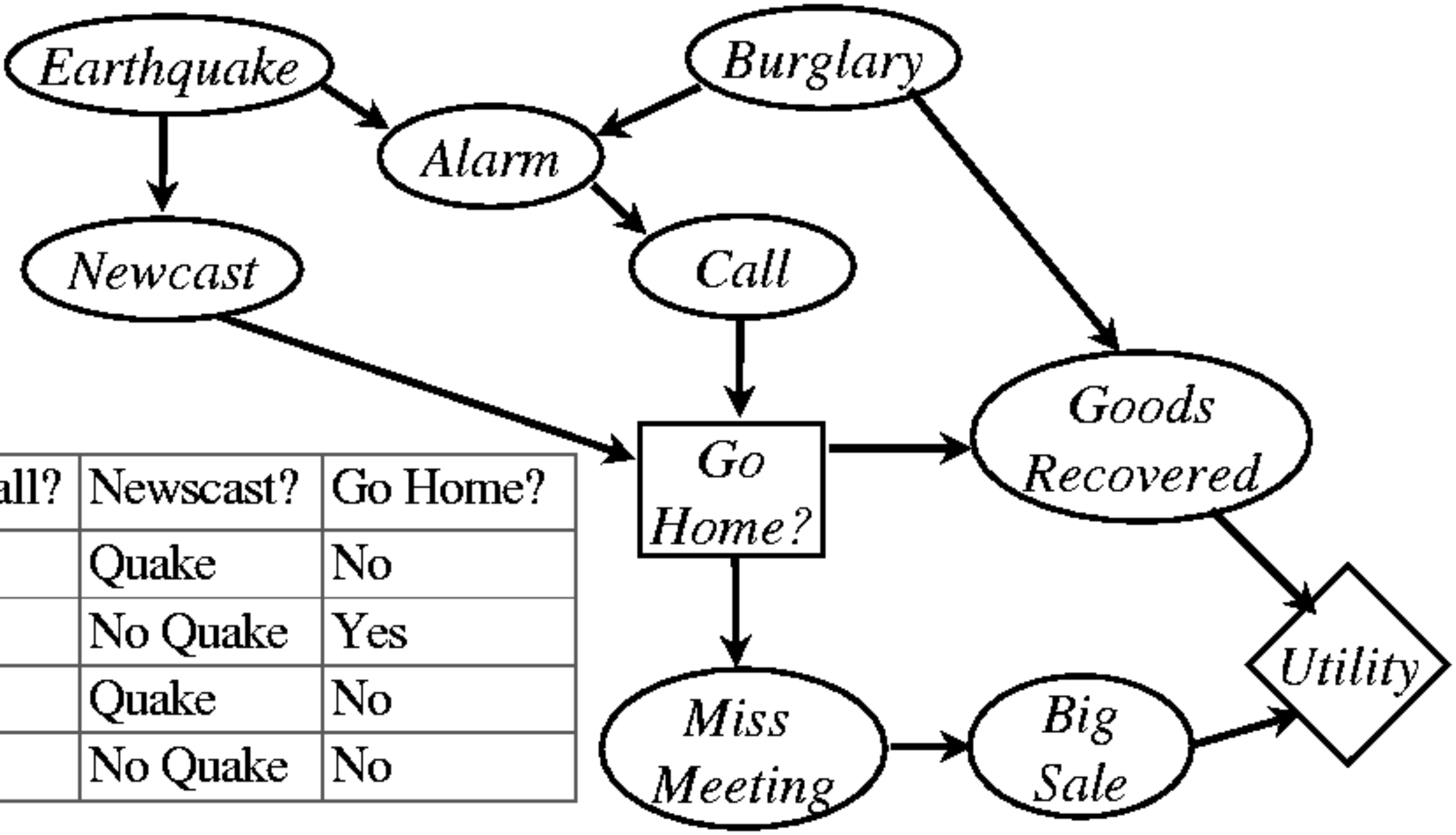
Expected Utility of this policy is 100

Value-of-Information in an Influence Diagram



How much better can we do when this arc is here?

Value-of-Information is the increase in Expected Utility



Phonecall?	Newscast?	Go Home?
Yes	Quake	No
Yes	No Quake	Yes
No	Quake	No
No	No Quake	No

Expected Utility of this policy is 112.5