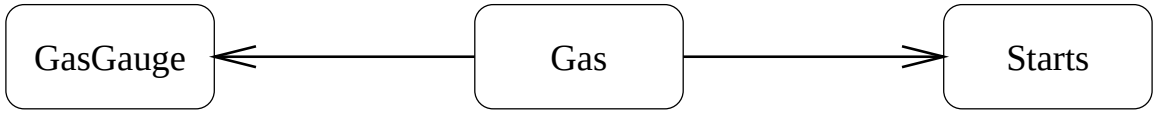
Inference in Belief Networks

Two Basic Algorithms

- **SPI**: Algebraic manipulation (Li & D'Ambrosio)
- **Junction Tree**: Message passing (Lauritzen & Spiegelhalter)

A Very Simple Belief Network



$\mathbf{P}(G)$	
Gas	.8
–Gas	.2

$\mathbf{P}(G\|M)$	Meter	–Meter
Gas	.9	.1
–Gas	.3	.7

$\mathbf{P}(S\|G)$	Starts	–Starts
Gas	.7	.3
–Gas	.01	.99

A Simple Example of SPI

What is the probability that there is gas in the tank (G) given that the gas meter (M) reads true?

- **What is desired probability distribution?** $\mathbf{P}(G|M)$
- **Apply definition of conditional probability**

$$\mathbf{P}(G|M) = \frac{\mathbf{P}(G, M)}{\mathbf{P}(M)}$$

- **Express each of the needed probabilities as a marginal of the full joint**

$$\mathbf{P}(G, M) = \sum_{S} \mathbf{P}(G, M, S)$$

$$\mathbf{P}(M) = \sum_{G,S} \mathbf{P}(G, M, S)$$

A Simple Example of SPI Continued

- **Compute each of these marginal distributions from the belief network decomposition**

$$\begin{aligned}\sum_S \mathbf{P}(G,M,S) &= \sum_S \mathbf{P}(G)\mathbf{P}(M|G)\mathbf{P}(S|G) \\ &= \mathbf{P}(G)\mathbf{P}(M|G)\sum_S \mathbf{P}(S|G) \\ &= \mathbf{P}(G)\mathbf{P}(M|G)\end{aligned}$$

$$\begin{aligned}\sum_{G,S} \mathbf{P}(G,M,S) &= \sum_{G,S} \mathbf{P}(G)\mathbf{P}(M|G)\mathbf{P}(S|G) \\ &= \sum_G \mathbf{P}(G)\mathbf{P}(M|G)\sum_S \mathbf{P}(S|G) \\ &= \sum_G \mathbf{P}(G)\mathbf{P}(M|G)\end{aligned}$$

- **Divide numerator by denominator**

$$\mathbf{P}(G|M) = \frac{\mathbf{P}(G)\mathbf{P}(M|G)}{\Sigma_G \mathbf{P}(G)\mathbf{P}(M|G)}$$

Simplifying the Method

- **Don't Compute Denominator**

  The denominator is just a normalizing factor = sum of all values of the numerator distribution.

  Just compute numerator distribution, then normalize.

- **Impose evidence by deleting entries in probability tables**

  Given that M is true, modify $\mathbf{P}(M|G)$ to become $\mathbf{P}[G]$

$\mathbf{P}(M\|G)$	Meter	–Meter
Gas	.9	.1
–Gas	.3	.7

$\Rightarrow$

$\mathbf{P}[G]$	
Gas	.9
–Gas	.3

- **Recursively delete all leaf nodes that are not the query and have no evidence**

  In this case, delete $\mathbf{P}(S|G)$, since S is not involved in the query and S has no evidence.

Resulting Procedure

- **Apply Evidence**: $\mathbf{P}(M|G) \Rightarrow \mathbf{P}[G]$
- **Determine Query Node**: G
- **Delete Leaves**: delete $\mathbf{P}(S|G)$
- **Compute conformal product of all remaining CPTs and sum over all remaining variables**:

$\mathbf{P}(G) \cdot \mathbf{P}[G] =$

$\mathbf{P}(G)$	
Gas	.8
–Gas	.2

$\cdot$

$\mathbf{P}[G]$	
Gas	.9
–Gas	.3

$=$

$\mathbf{P}[G]$	
Gas	.72
–Gas	.06

No other variables to sum over.

- **Normalize** (normalizing factor is 0.78)

$\mathbf{P}[G]$	
Gas	.923
–Gas	.077

Continuing the Example

Suppose we now observe that the car starts. What is $\mathbf{P}(G|M,S)$?

- **Apply Evidence:** $\mathbf{P}(M|G) \Rightarrow \mathbf{P}[G]_1$, $\mathbf{P}(S|G) \Rightarrow \mathbf{P}[G]_2$
- **Determine Query Node:** G
- **Delete Leaves**: None
- **Conformal Product of Remaining Nodes:**

$\mathbf{P}(G) \cdot \mathbf{P}[G]_1 \cdot \mathbf{P}[G]_2 =$

$\mathbf{P}(G)$	
Gas	.8
–Gas	.2

$\cdot$

$\mathbf{P}[G]_1$	
Gas	.9
–Gas	.3

$\cdot$

$\mathbf{P}[G]_2$	
Gas	.7
–Gas	.01

$=$

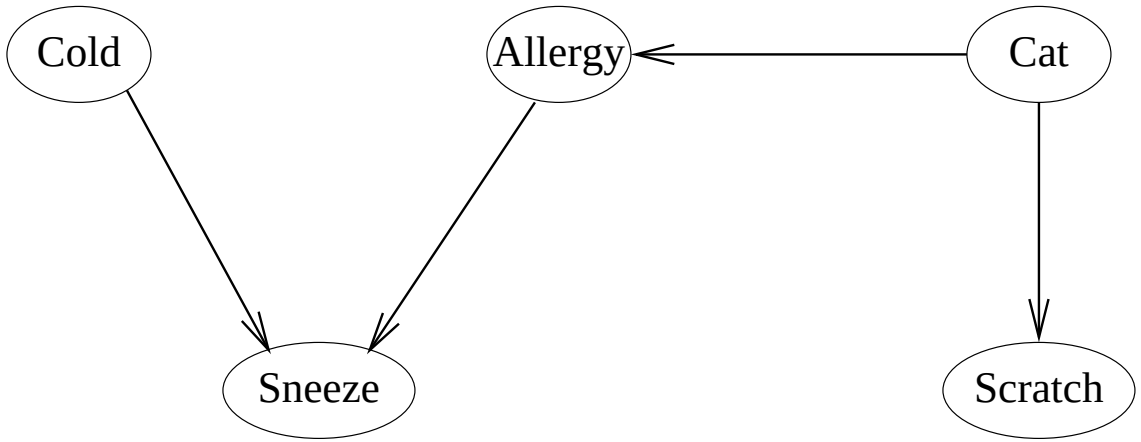
$\mathbf{P}[G]$	
Gas	.5040
–Gas	.0006

No other variables to sum over.

- **Normalize** (normalizing factor is 0.5046)

$\mathbf{P}(G)$	
Gas	.9988
–Gas	.0012

A More Complex Example



What is $P(Cold|Sneeze)$?

SPI for Colds

- **Apply Evidence**: $\mathbf{P}(Sneeze|Cold, A) \Rightarrow \mathbf{P}[Cold, A]$
- **Determine Query Node**: $Cold$
- **Delete Leaves**: delete $\mathbf{P}(Scratch|Cat)$
- **Conformal Product and sum over remaining variables**:

$$\sum_{A,Cat} \mathbf{P}(Cold) \cdot \mathbf{P}[Cold, A] \cdot P(A|Cat) \cdot P(Cat) \; =$$
$$\mathbf{P}(Cold) \cdot \sum_{A} \mathbf{P}[Cold, A] \cdot \sum_{Cat} P(A|Cat) \cdot P(Cat)$$

- **Normalize**

Greedy Algorithm for choosing the elimination order

$Nodes$ = set of tables

$Vars$ = set of variables to sum over

while $|Nodes| > 1$ **do**

Generate all pairs of tables in $Nodes$ that share at least one variable

Compute size of table that would result from conformal product of each pair

(summing over as many variables V as possible)

Let (T_1, T_2) be the pair with the smallest resulting size (sum over V)

Delete T_1 and T_2 from $Nodes$

Add conformal product $\Sigma_V T_1 \cdot T_2$ to $Nodes$

end

Example of Greedy Algorithm

Given Tables: $\mathbf{P}(Cold), \mathbf{P}[Cold, A], \mathbf{P}(A|Cat), \mathbf{P}(Cat)$

Variables to Sum: A, Cat

Candidate pair	Size of product	Sum over	Size of result
$\mathbf{P}(Cold) \cdot \mathbf{P}[Cold, A]$	2	nil	2
$\mathbf{P}[Cold, A] \cdot \mathbf{P}(A\|Cat)$	3	A	2
$\mathbf{P}(A\|Cat) \cdot \mathbf{P}(Cat)$	2	Cat	1

Choose: $\mathbf{P}[A] = \sum_{Cat} \mathbf{P}(A|Cat) \cdot \mathbf{P}(Cat)$

Given Tables: $\mathbf{P}(Cold), \mathbf{P}[Cold, A], \mathbf{P}[A]$

Variables to Sum: A

Candidate Pair	Size of product	Sum over	Size of result
$\mathbf{P}(Cold) \cdot \mathbf{P}[Cold, A]$	2	nil	2
$\mathbf{P}[Cold, A] \cdot \mathbf{P}[A]$	2	A	1

Choose: $\mathbf{P}[Cold] = \sum_{A} \mathbf{P}[Cold, A] \cdot \mathbf{P}[A]$

Given Tables: $\mathbf{P}(Cold), \mathbf{P}[Cold]$

Variables to Sum: none

Only one candidate: $\mathbf{P}[Cold] = \mathbf{P}(Cold) \cdot P[Cold]$

Full SPI Algorithm

GenerateCandidates($table2$)

    for each $table$ **in** $candidates$

        if $table$ and $table2$ share one or more variables

            create new candidate nc that will multiply $table$ and $table2$

            add nc to $candidates$

    end for

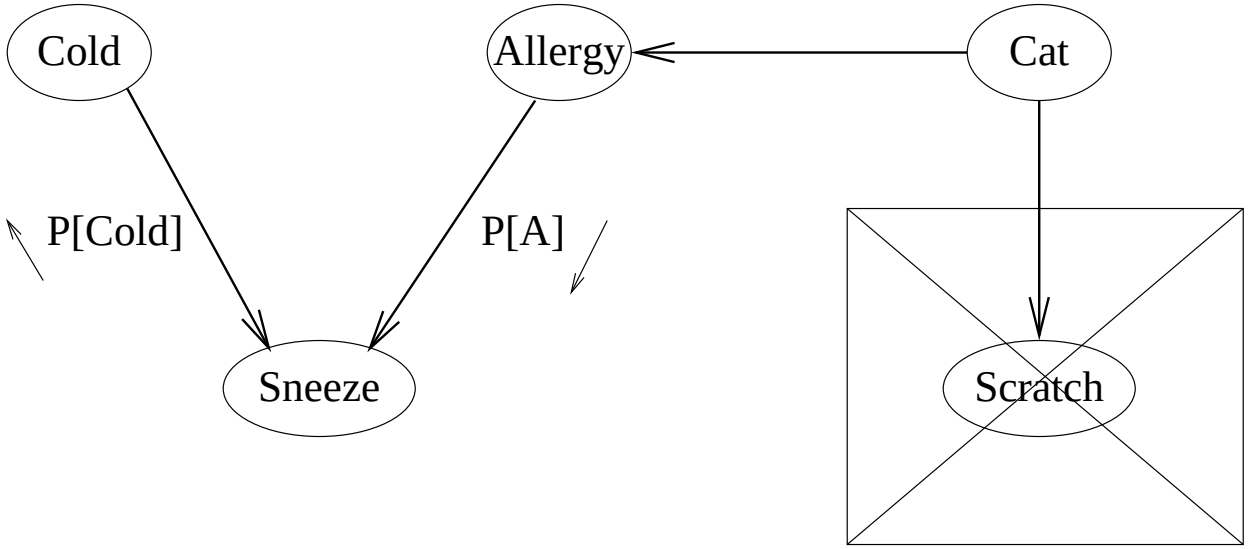
end GenerateCandidates

Full SPI Algorithm (2)

SPI($net, evidence, query$) **returns** *table*
net: list of probability tables
evidence: list of (variable value) pairs
query: list of variables

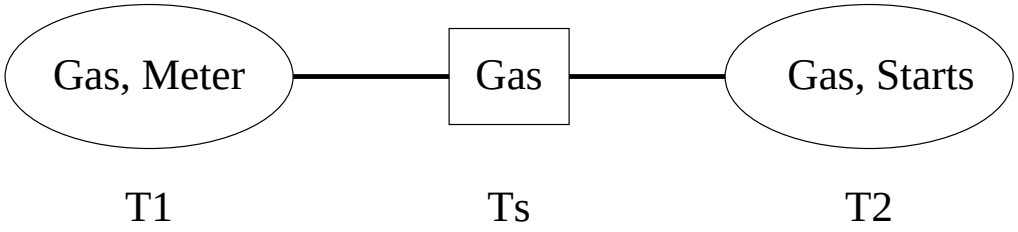
$vsum$ = all variables − (evidence variables ∪ query variables)
$candidates$ = empty set
for *table* **in** *net* **do**
apply *evidence* (if any) to *table*
let $margvars$ = legal marginalizers of *table* in $vsum$
$table'$ = MARGINALIZE($table, margvars$)
GENERATECANDIDATES($table'$)
end for
while $|candidates| > 0$ **do**
c = best *candidate* (candidate with smallest result table)
$margvars$ = legal marginalizers of c
delete $c.left$ and $c.right$ from $candidates$
delete all c' from $candidates$ if c' involves either $c.left$ or $c.right$
$table' = c.left.table \cdot c.right.table$
$table'$ = MARGINALIZE($table, margvars$)
if ($|candidates| == 0$)
$table'$ = NORMALIZE($table'$)
return $table'$
else
GENERATECANDIDATES($table'$)
end while
end SPI

Viewing SPI as Message Passing



Every time we sum over a variable, we can view the resulting table as a "message" that is being passed around the network. We will see that this is the idea underlying the Junction Tree algorithm.

The Junction Tree Representation



A junction tree is an undirected tree with two kinds of nodes: Cluster nodes and separator nodes. It has the following properties:

- **Every random variable belongs to one or more cluster nodes and may belong to one or more separator nodes**
- **Every separator node has exactly two cluster-node neighbors** (it may help to think of the separator nodes as being information attached to the edge connecting two cluster nodes).
- **The random variables in a separator node are the set intersection of the random variables in the two cluster nodes that it connects**
- **If a variable belongs to two cluster nodes, it belongs to all cluster nodes on the path joining them.** (The running intersection property)

Contents of the Junction Tree Nodes

- **Each cluster node contains a probability table**

  $\mathbf{P}[Gas, Meter] = \mathbf{P}(Gas) \cdot \mathbf{P}(Meter|Gas)$

  $\mathbf{P}[Gas, Starts] = \mathbf{P}(Starts|Gas)$

- **Each separator node contains a probability table**

  Initially, the table is full of 1's.:

  $\mathbf{P}[Gas] = (1.0, 1.0)$

To build these tables:

Initialize all tables to 1.0

For each CPT in the belief net, choose one cluster node that contains all of its variables

Multiply it into the table of the cluster node

Conformal Quotients

Given two tables, we can divide one by the other:

$\mathbf{P}(G, M)$	Meter	–Meter
Gas	.72	.08
–Gas	.06	.14

$\div$

$\mathbf{P}(G)$	
Gas	.8
–Gas	.2

$=$

$\mathbf{P}(M\|G)$	Meter	–Meter
Gas	.72/.8	.08/.8
–Gas	.06/.2	.14/.2

$=$

$\mathbf{P}(M\|G)$	Meter	–Meter
Gas	.9	.1
–Gas	.3	.7

We cannot divide by zero, but $0 \div 0 = 0$ is ok.

Junction Tree Invariant

The joint distribution is the conformal product of the cluster nodes divided by the conformal product of the separator nodes

$$P(G, M, S) = \frac{P[G, M] \cdot P[G, S]}{P[G]}$$

We will see that the separator nodes store information about the "history" of the updates that have been performed to each node. By dividing by this history information, we can "undo" the updates.

In our initial construction, each original table from the belief net was multiplied into *one* of the cluster nodes, and the separator nodes are all 1.0, so this invariant is trivially satisfied.

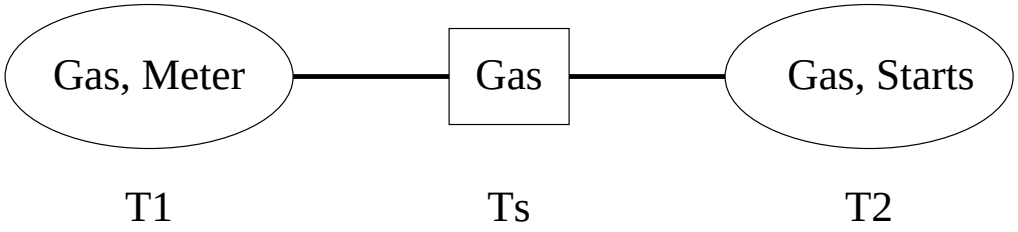
Junction Tree Inference

Inference consists of three phases:

- **Message Passing:** Evidence is entered into the network as a message, and then messages are passed around the tree.
- **Marginalization:** Choose a node N that contains your query variable, V, and compute $\mathbf{P}[V] = \Sigma_{N-V}\, \mathbf{P}[N]$.
- **Normalization:** Normalize $\mathbf{P}[V]$.

During message passing, every cluster node receives one incoming message from each separator and sends an outgoing message to each separator. Message passing is complete when each separator has passed a message in each direction. At this point, the tree is said to be *consistent*.

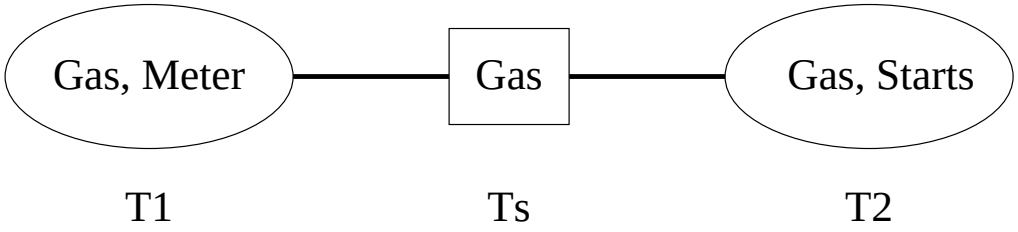
Passing a Message: Absorption



To pass a message from $(Gas, Meter)$ to $(Gas, Starts)$, we perform the following:

- **Marginalize T_1 wrt separator variables:** $T_s^* = \Sigma_M \, \mathbf{P}[G, M]$
- **Update:** $T_2 := T_2 \cdot (T_s^* / T_s)$
- **Update the separator table:** $T_s := T_s^*$

Passing a Message: Example



In our example:

- **Marginalize T_1:** $T_s^* =$

$\mathbf{P}(G)$	
Gas	.8
–Gas	.2

- **Update T_2:** $P(S,G) = P(S|G) \cdot (P(G)/\mathbf{1}) =$

$\mathbf{P}(S,G)$	Starts	–Starts
Gas	.56	.24
–Gas	.002	.198

- **Update T_s:** $T_s = T_s^* =$

$\mathbf{P}(G)$	
Gas	.8
–Gas	.2

So the three tables are $\mathbf{P}(G,M), \mathbf{P}(G), \mathbf{P}(G,S)$. We can compute $P(M)$ by marginalizing T_1, $P(G)$ by T_s, and $P(S)$ by marginalizing $P(G,S)$.

Passing a Message (3)

Things to notice:

- Our original $\mathbf{P}(G)$ table has now been multiplied into *both* cluster nodes. But the invariant still holds:

$$\mathbf{P}(G, M, S) = \frac{\mathbf{P}(M|G) \cdot \mathbf{P}(G) \cdot \mathbf{P}(S|G) \cdot \mathbf{P}(G)}{\mathbf{P}(G)}$$

- If we had done the propagation from T_2 to T_1, we would have computed

$$\begin{aligned} T_s^* &:= \sum_S \mathbf{P}(S|G) = (1,1) \\ T_1 &:= T_1 \cdot [(1,1)/(1,1)] = T_1 \\ T_s &:= (1,1) \end{aligned}$$

  This would have left the junction tree unchanged.

- After propagating from T_1 to T_2, if we do the reverse propagation what happens?

$$\begin{aligned} T_s^* &:= \sum_S \mathbf{P}(S, G) = \mathbf{P}(G) \\ T_1 &:= T_1 \cdot (\mathbf{P}(G)/\mathbf{P}(G)) = T_1 \\ T_s &:= \mathbf{P}(G) \end{aligned}$$

  So this also leaves the junction tree unchanged.

Recording Evidence

Suppose we observe M is true and we want $P(G|M)$.

- **Find a cluster node containing M**

$T_1 = \mathbf{P}(G, M) =$

$\mathbf{P}(G, M)$	Meter	–Meter
Gas	.72	.08
–Gas	.06	.14

- **Zero out all entries of that table that disagree with the evidence**

$\mathbf{P}[G, M]$	Meter	–Meter
Gas	.72	0
–Gas	.06	0

- **Perform Propagation**
- **Choose node containing G**
- **Marginalize and Normalize**

Propagation of the Evidence

- **Marginalize T_1:**

$$T_s^* = \Sigma_M \, \mathbf{P}[G, M] =$$

	$\mathbf{P}[G]$
Gas	.72
–Gas	.06

- **Update T_2:**

$$T_2 := T_2 \cdot (T_s^* / T_s) =$$

$\mathbf{P}(S, G)$	Starts	–Starts
Gas	.5040	.2160
–Gas	.0006	.0594

- **Update $T_s := T_s^*$**

The reverse message causes no changes.

Extracting the Answer

We can choose any node containing G, even a separator node, such as $\mathbf{P}[G]$:

$\mathbf{P}[G]$

Gas	.72
–Gas	.06

Normalization gives the answer:

$\mathbf{P}[G]$

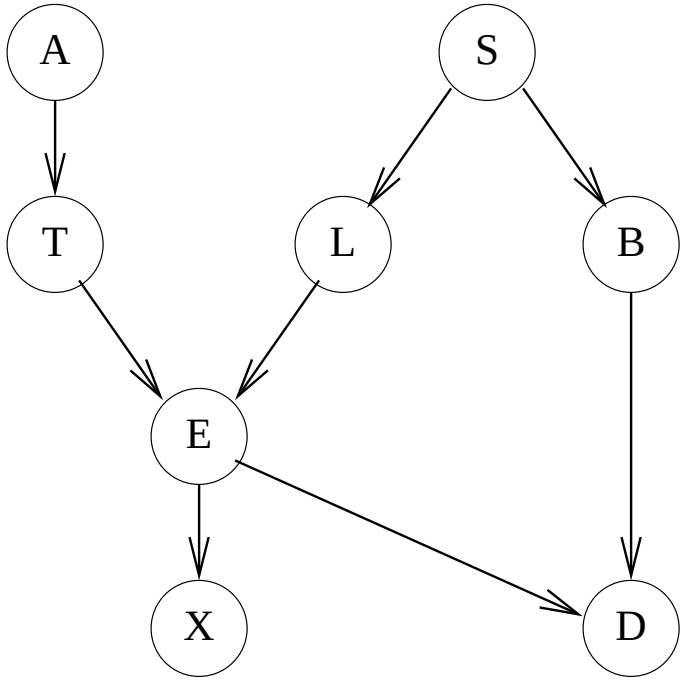
Gas	.923
–Gas	.077

Graphical Method of Building the Junction Tree

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

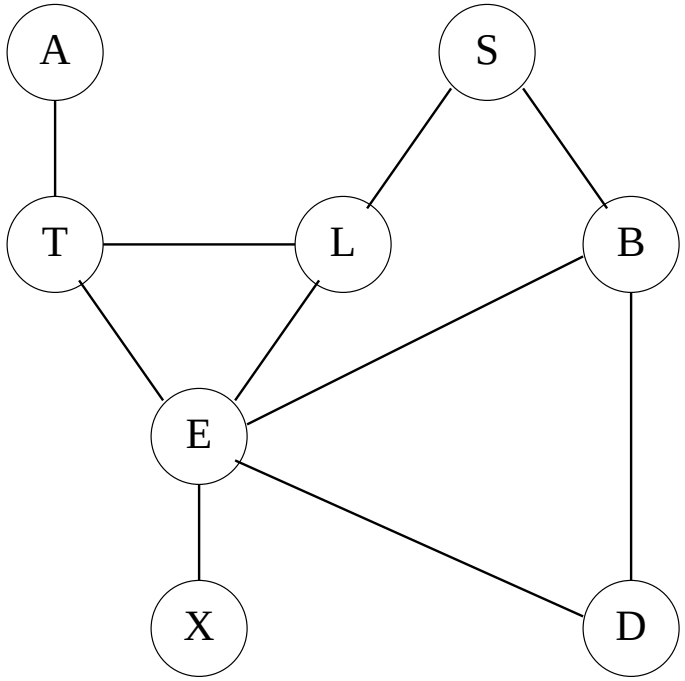
- **Marry the Parents** ("moralize the graph"): Add an undirected edge between every pair of parents of a node (unless they are already connected.
- **Make All Arrows Undirected**
- **Triangulate the Graph:** Add edges so that every cycle of length 4 or more contains a chord.
- **Identify the maximal 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: ASIA



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

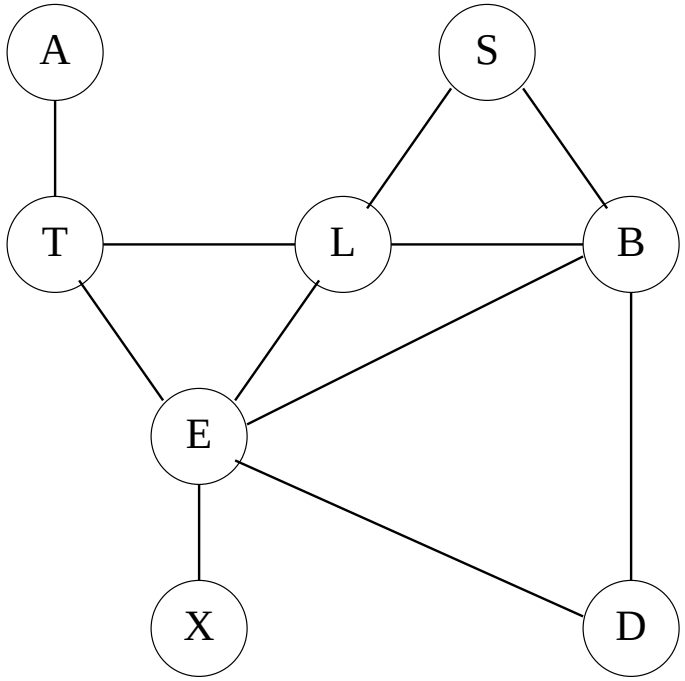
Step 1: Moralize the Graph



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

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

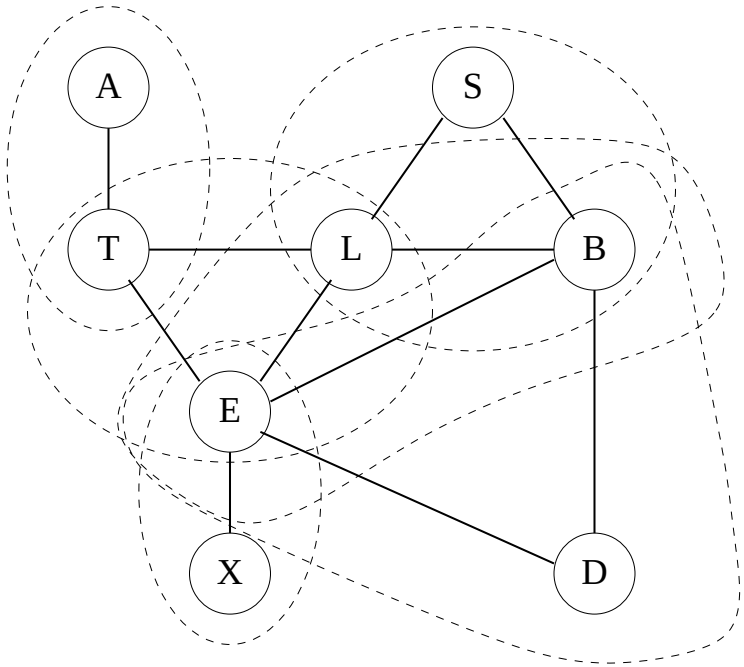
Step 2: Triangulate the Moral Graph

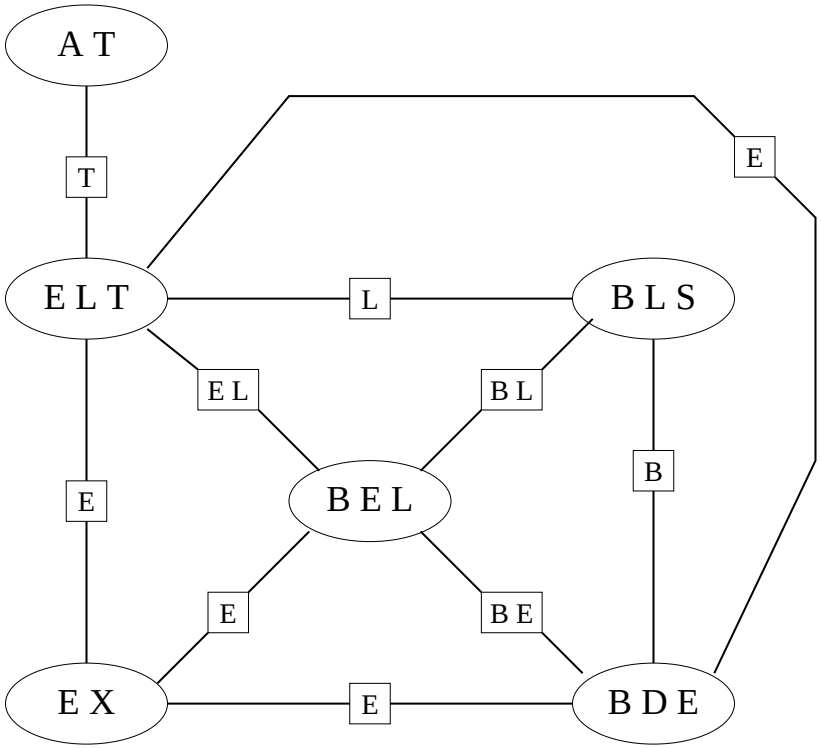


There is a cycle of length four with no shortcuts: **E**, **L**, **S**, **B**.

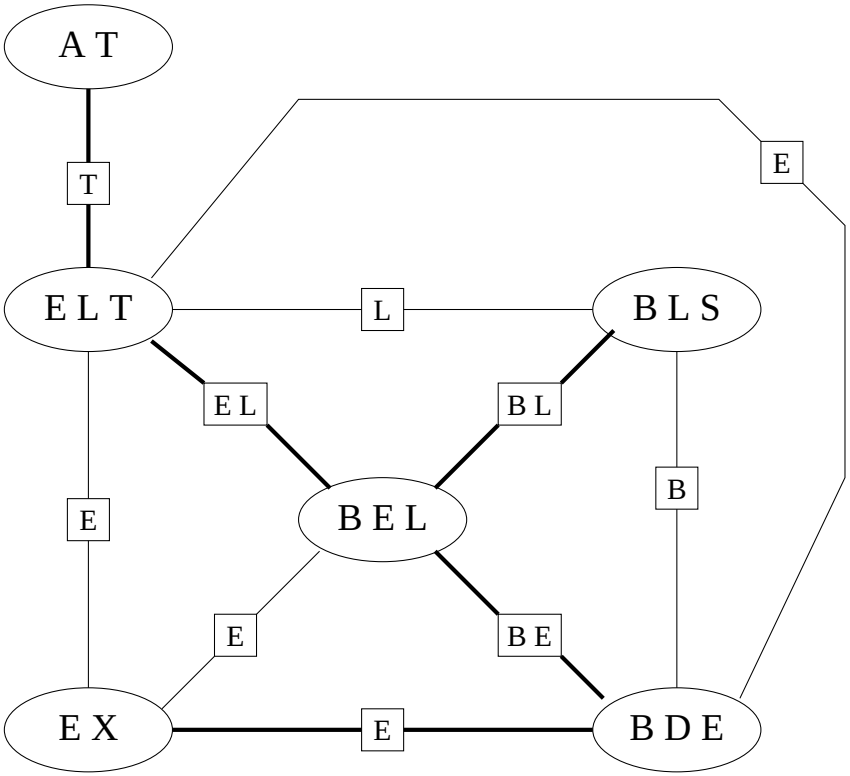
We have a choice of where to add the shortcut. Either **LB** or **SE** would work.

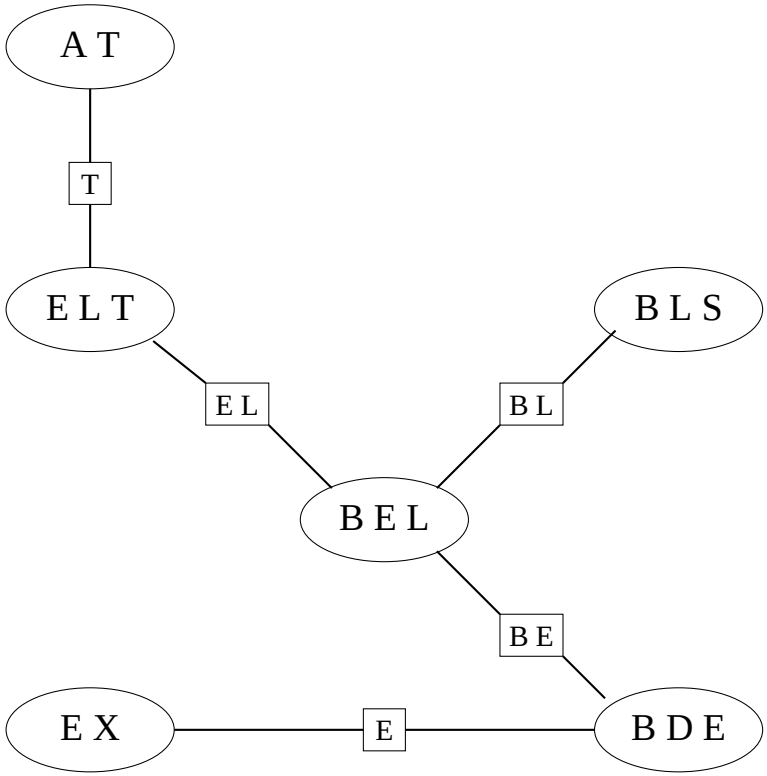
Step 3: Cliques and Junction Graph





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).

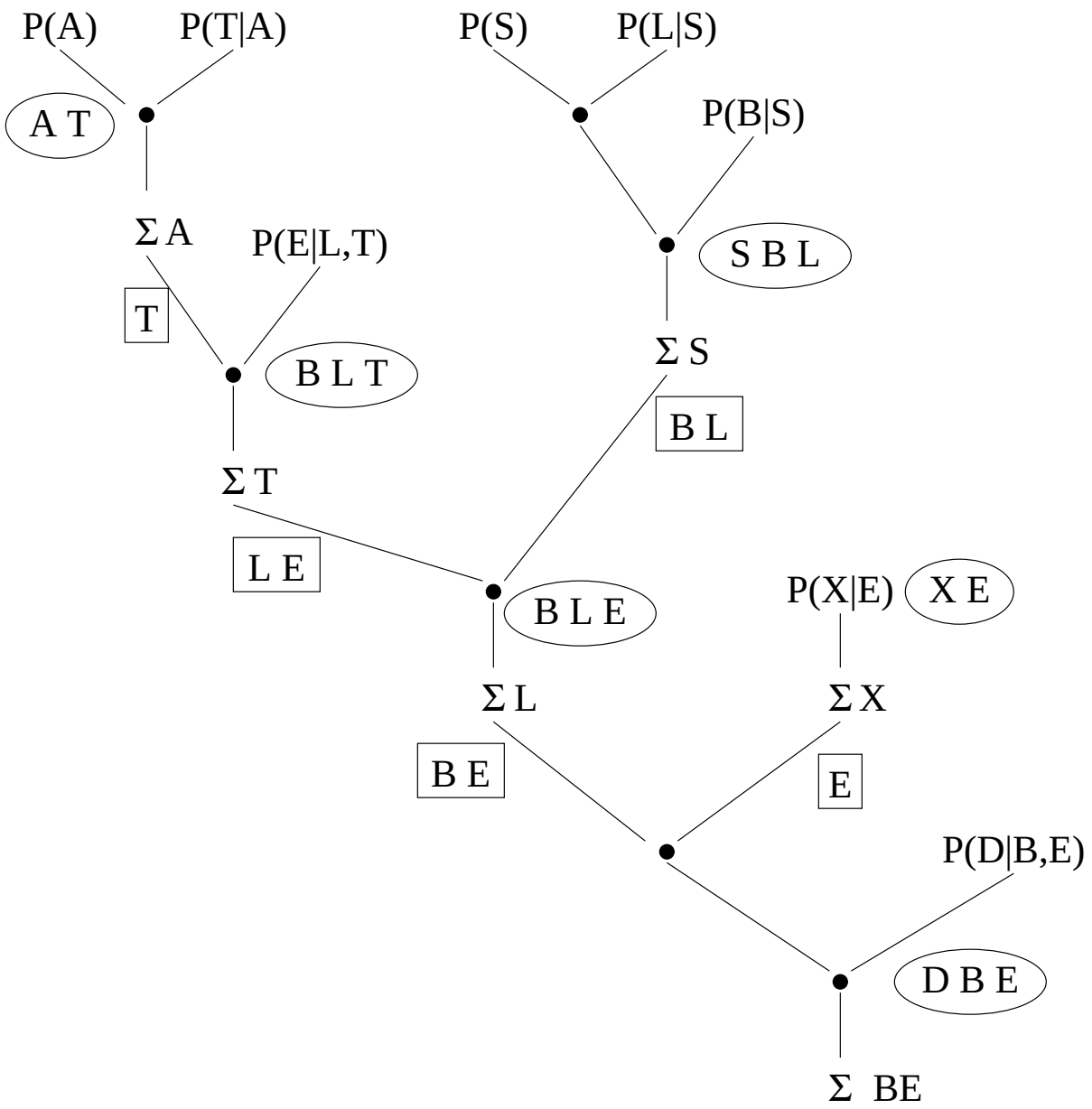
Junction Trees and Elimination Orders (D'Ambrosio)

Suppose we want to compute the marginal probability of one node (e.g., **D**). We can view the junction tree as defining one order in which we marginalize away the irrelevant variables: A, T, S, L, X, B, E

$$
\begin{aligned}
P(D) &= \sum_{A,T,S,L,X,B,E} P(A)P(S)P(T|A)P(L|S)P(B|S)P(E|L,T)P(D|B,E)P(X|E) \\
&= \sum_{T,S,L,X,B,E} P(S)P(L|S)P(B|S)P(E|L,T)P(D|B,E)P(X|E) \sum_{A} \Big[P(A)P(T|A)\Big] \\
&= \sum_{S,L,X,B,E} P(S)P(L|S)P(B|S)P(D|B,E)P(X|E) \sum_{T} \Big[P(E|L,T) \sum_{A} P(A)P(T|A)\Big] \\
&= \sum_{L,X,B,E} P(D|B,E)P(X|E) \sum_{S} \Big[P(S)P(L|S)P(B|S)\Big] \sum_{T} P(E|L,T) \sum_{A} P(A)P(T|A) \\
&= \sum_{X,B,E} P(D|B,E)P(X|E) \sum_{L} \Big[\sum_{S} P(S)P(L|S)P(B|S) \sum_{T} P(E|L,T) \sum_{A} P(A)P(T|A)\Big] \\
&= \sum_{B,E} \Big[P(D|B,E) \sum_{X} \Big[P(X|E)\Big] \sum_{L} \sum_{S} P(S)P(L|S)P(B|S) \sum_{T} P(E|L,T) \sum_{A} P(A)P(T|A)\Big]
\end{aligned}
$$

The separator tables along the way are the results of each $\sum$.

Expression Graph Showing the Elimination Order



Relationship of SPI and Junction Tree

In SPI, the elimination order is recomputed for each query. Efficiency is gained by caching previously-computed intermediates.

In JT, the graph represents a sharable elimination order.

Implementing Junction Tree using SPI

Based on D'Ambrosio's insight that JT = SPI for some elimination ordering, we can simply use SPI to compute a good elimination order for the joint distribution, and this will construct the junction tree. (It is much easier than all of that graph manipulation stuff.)

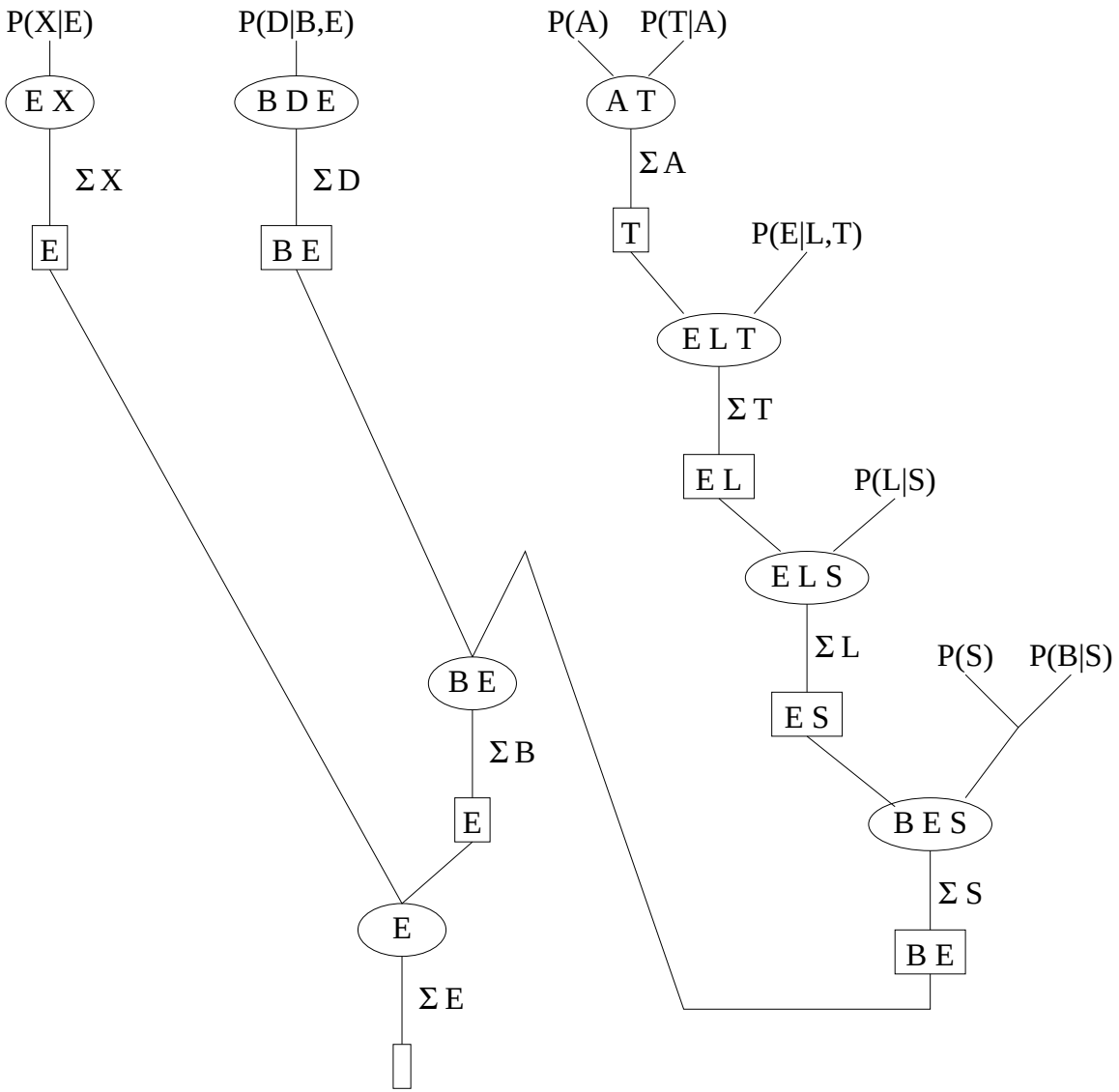
The elimination order for Asia computed by SPI is

$$X, D, A, T, L, S, B, E$$

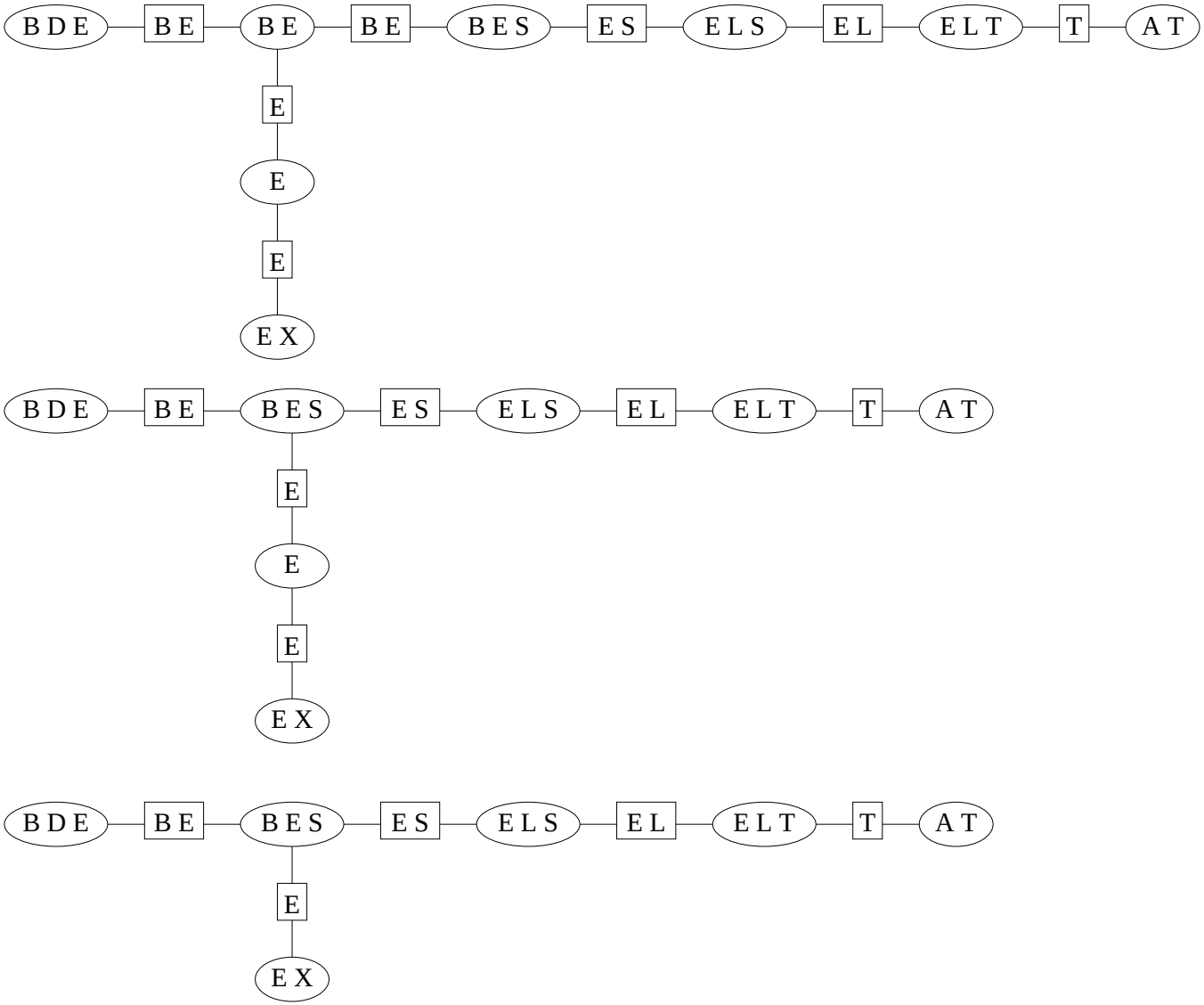
The expression is

$$\sum_{B,E}[\sum_{S}[P(S)P(B|S)][\sum_{L}P(L|S)[\sum_{T}P(E|L,T)[\sum_{A}P(A)P(T|A)]]]][[\sum_{D}P(D|B,E)][\sum_{X}P(X|E)]]$$

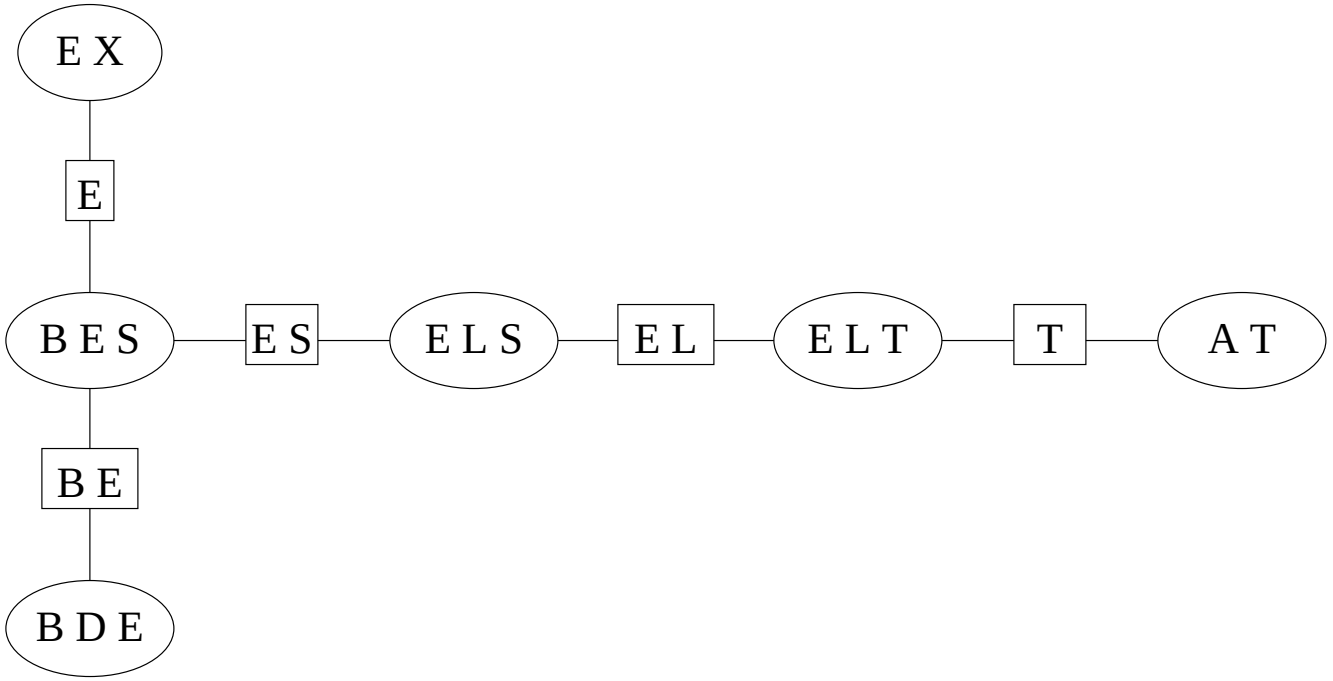
Expression Graph



Deleting Useless Nodes



Junction Tree



Algorithm Details

procedure BUILDJT(belief net)
let T be the current list of tables (either original tables or separator tables)
initially T := all original tables from belief net
repeat
    choose next variable V to eliminate (e.g., using SPI heuristic)
    let S = all tables from T involving V; remove them from T
    create new cluster node N containing all variables of tables in S
    let table of N = conformal product of all original tables in S
    for each derived table in S add a link from N to their source node
    let R be a new separator table whose variables are the variables of N
    set the cluster node of R to be N
    remove V from R and add R to T
until R is an empty table
for each separator node S with left cluster node L and right cluster node R
        if the variables in S == the variables in L
            delete S and merge L into R (including its links)
        else if the variables in S == the variables in R
            delete S and merge R into L (including its links)
end for

Algorithm Details (2)

Variable Eliminated	Conformal Product	Cluster Node Created	Links Created	Separator Node Created
X	$P(X\|E)$	XE	nil	$P[E]_1$
D	$P(D\|B,E)$	BDE	nil	$P[B,E]_1$
A	$P(A)P(T\|A)$	AT	nil	$P[T]$
T	$P(E\|L,T)P[T]$	ELT	AT	$P[E,L]$
L	$P(L\|S)P[E,L]$	ELS	ELT	$P[E,S]$
S	$P(S)P(B\|S)P[E,S]$	BES	ELS	$P[B,E]_2$
B	$P[B,E]_1P[B,E]_2$	BE	BDE,BES	$P[E]_2$
E	$P[E]_1P[E]_2$	E	XE,BE	nil

Postprocessing to remove useless nodes:

We have BE – $P[BE]$ – BES, so we merge BE into BES.

BES becomes linked to E and BDE.

We have XE – $P[E]$ – E, so we merge E into XE.

XE becomes linked to BES.