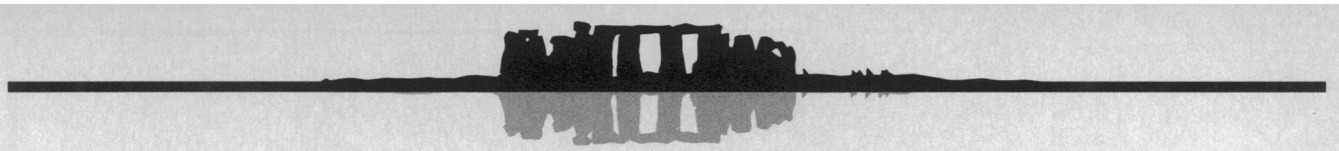


Propagating Belief Functions with Local Computations

Prakash P. Shenoy and Glenn Shafer
University of Kansas

In this article, we describe a way to propagate belief functions in certain kinds of trees using only local computations. This scheme generalizes the computational scheme proposed by Shafer and Logan[1] for diagnostic trees of the type studied by Gordon and Shortliffe[2,3] and the slightly more general scheme proposed by Shafer[4] for hierarchical evidence. It also generalizes the computational scheme proposed by Pearl[5] for Bayesian causal trees.

Pearl's causal trees and Gordon and Shortliffe's diagnostic trees are both ways of breaking down the evidence that bears on a large problem into smaller items of evidence that bear on smaller parts of the problem so that these smaller problems can be dealt with one at a time. This localization of effort is often essential to make the process of probability judgment feasible, both for the person who is making probability judgments and for the machine that is combining them. The basic structure for our scheme is a type of tree that generalizes both Pearl's and Gordon and Shortliffe's trees. Trees of this type permit localized computation in Pearl's sense. They are based on qualitative judgments of conditional independence.

We believe that the scheme we describe here will prove useful in expert systems. It is now clear that the successful propagation of probability or certainty factors in expert systems requires much more structure than can be provided in a pure production-system framework. Bayesian schemes, on the other hand, often make unrealistic demands for structure. The propagation of belief functions in trees and more general networks occupies a middle ground where some sensible and useful things can be done.

We would like to emphasize that the basic idea of local computation for propagating probabilities is Judea Pearl's. It is an innovative idea; we do not believe that it can be found in the Bayesian literature prior to Pearl's work. We see our contribution as extending Pearl's idea from Bayesian probabilities to belief functions.

We will describe the scheme proposed by Pearl[5] for Bayesian causal trees. Then, we will describe the scheme proposed by Shafer and Logan[1] for diagnostic trees, and, afterwards, as a background to our own scheme, we will describe qualitative Markov trees. Finally, we will describe our belief-function scheme, which has Pearl's and Shafer and Logan's schemes as special cases.

Pearl's scheme for Bayesian causal trees

A Bayesian causal tree is a directed tree—i.e., a graph in which the links are directed, each node has only one incoming link, and there are no cycles. A typical node, say B, represents a random variable with a finite number of possible values, say $b_1, \ldots, b_n$. If B is the answer to a theoretical question, then the values are hypotheses; if B is an observable, then the values are possible observations. It is assumed that the variables in the tree have a joint probability distribution, and the structure of the tree reflects assumptions about this distribution. If B separates A and C (as in Figure 1), this is interpreted to mean that A and C are conditionally in-

0885-9000/86/0800-0043 $01.00

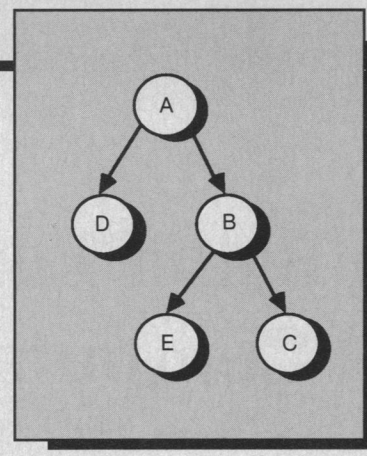


Figure 1. An example of a causal tree (left).

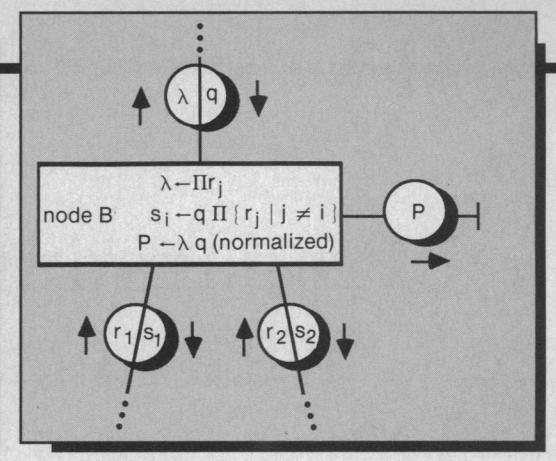


Figure 2. A typical node processor with two daughters (right).

dependent given B with respect to this distribution. In particular, daughter nodes are conditionally independent given their mother. Intuitively, a link from A to B means that A directly causes or influences B.

Conceptually, a prior joint probability distribution for all the variables in a Bayesian causal tree is determined by a prior probability distribution for the topmost (initial) node and transition probabilities from each node to its daughters. This prior joint distribution can be updated to a posterior distribution when values for particular variables are observed. It may be prohibitively difficult to actually compute the prior and posterior joint distributions. It is clear, however, that the prior distributions for the individual nodes can be obtained by working down the tree, step-by-step. This step-by-step process obviously involves only local computations; we go from each node to its daughters, without having to consider the joint distribution for nodes that are widely separated in the tree. Pearl has shown that the posterior distributions for the individual nodes can be obtained by a similar process, one that similarly propagates information through the tree using only local computations.

In the following description of Pearl's computational scheme we assume, for simplicity, that the variables whose values we observe are represented by terminal nodes.

Processors. There is a processor at each node of the tree, and also at each link. Each node stores (or is prepared to produce on demand) the current probability distribution for its variable. Each link stores the probability transition matrix connecting its two variables—the matrix that gives probabilities for the lower variable conditional on values of the upper variable.

Typical node processor. Each node has input-output ports for each neighboring node and also an output port for reporting its current probability distribution P. Figure 2 shows the action of the processor for a typical node, which we denote by B. The symbols P, λ, q, r_j, and s_i in this figure all represent column vectors. These vectors all have the same dimension, equal to the number of possible values of the variable represented by the node B. Multiplication of these vectors is component-wise. The arrows represent the direction of data flow. The vectors r_j and λ represent what statisticians call likelihoods—conditional probabilities of the evidence given each possible value of the variable represented by B. Here the evidence consists of observed values of some of the terminal nodes. The ith component of λ is the conditional probability of all the observed values below B given that the actual value of B is b_i. The ith component of r_j is the conditional probability of all the observed values at or below the jth daughter of B given that the actual value of B is b_i. Since the daughter nodes are independent given the mother node, the likelihood vector λ is simply the product of the likelihood vectors r_j.

The vectors q and s_i represent (non-normalized) posterior marginal probability distributions for node B, i.e., some constant times the vector $(P(b_1 \mid E), \ldots, P(b_n \mid E))$, where E denotes the evidence accounted for so far. The vector q represents the marginal probability distribution of node B posterior only to the evidence coming from the node set complement to the set of nodes at or below node B. It is not necessary to normalize the vectors q and s_i so that their components add to one. We do want to normalize the vector P, however, since it will go outside the system and be interpreted as a vector of probabilities. The vector s_i represents the marginal probability distribution of node B posterior only to the evidence coming from the node set complement to the set of nodes at or below the ith daughter of node B. The formulae for the s_i's and P in Figure 2 result from Bayes's rule and the fact that the evidence coming from the nodes at or below node B and the evidence coming from the complementary set of nodes are conditionally independent given node B.

As Figure 3 shows, the initial (topmost) node and the terminal nodes are slightly different. In particular, the vector q entering the initial node from above and the vector r entering a terminal node from below are inputs from outside the system.

Typical link processor. In Figure 4, the matrix M, which is stored permanently at the link, has entries $M_{i,j} = Pr(b_j \mid a_i)$. The multiplications at the link processors are matrix multiplications. The symbol M^t denotes the transpose of the matrix M.

Timing. A processor does nothing so long as it is in equilibrium—i.e., the contents of its output ports match the values calculated from the formulae using the contents of its input ports. If there is a mismatch, then the processor will eventually update its outputs. It does not matter whether the updating occurs regularly or at random intervals in time.

Initial values. Just for fun, let us initially put $q = (0, \ldots, 0)$ into the initial node and $r = (1, \ldots, 1)$ into the terminal nodes. This results in an equilibrium with all $r = s = \lambda = (1, \ldots, 1)$ and all $q = (0, \ldots, 0)$ (of the appropriate dimensions) though the normalization required to produce the out-

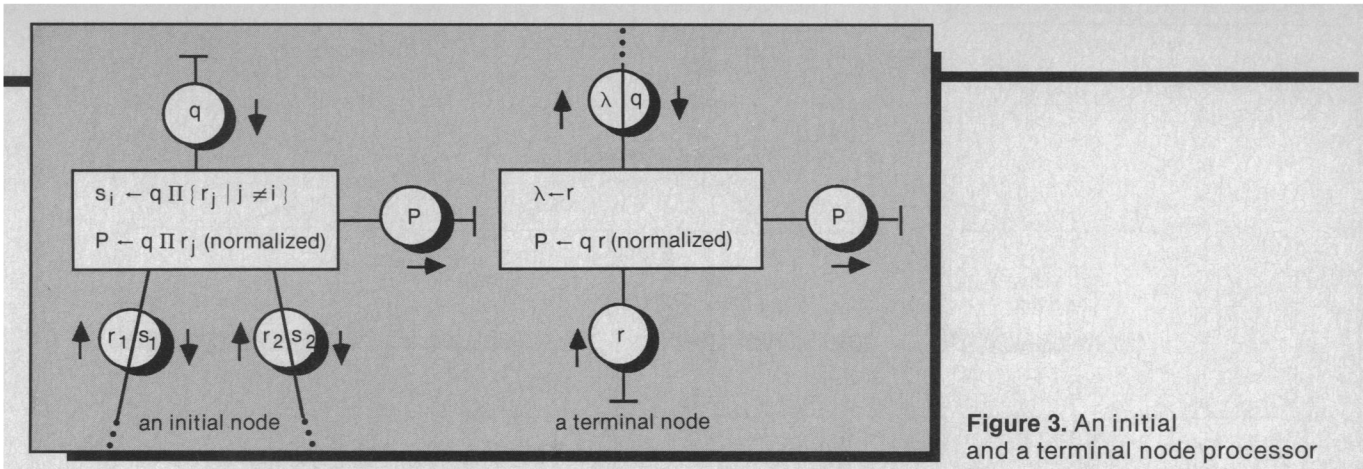


Figure 3. An initial and a terminal node processor

put probabilities is impossible. Now change q at the topmost node to a vector of initial probabilities for the variable represented by that node. This will propagate through the network to result in corresponding initial probabilities for all the other nodes.

Instantiation of terminal nodes. Whenever the jth state of a terminal node is observed to be true, we change its input to $r = (0, \ldots, 0, 1, 0, \ldots, 0)$, where the 1 occurs in the jth position.

Pearl's scheme fuses and propagates the effect of new evidence and beliefs through the causal tree in such a way that, when equilibrium is reached, each proposition will be assigned a probability consistent with the axioms of probability theory. As a whole, the system has as inputs the prior probability q for the initial node and the likelihoods r for the terminal nodes. As outputs, it has a probability distribution P for each node. Notice that the r's and λ's are not affected by changes in the q's and s's. Changes in the r's and λ's are communicated upwards only. They do, however, cause changes in q's and s's, which are communicated downwards. Input of a new q to the initial node will therefore result in equilibrium after one pass down the tree. An input of new r's to terminal nodes will cause a change in the r's and λ's propagating up the tree; from each node reached on the path up the tree, we will then have changes in the P's, q's and s's propagating back down the tree. So equilibrium will be reached in the time it takes to go up the tree and back down. It is clear that the values of λ and r throughout are determined by the values of r input at the terminal nodes. Once these values are fixed, the values of P, q, and s throughout are determined by the value of q input at the initial node. So all the equilibrium values are uniquely determined by the input values.

Belief functions in diagnostic trees

Gordon and Shortliffe[2,3] discuss the problem of pooling evidence in the case where all that evidence is focused for or against subsets of a frame of discernment that can be arranged hierarchically in a tree. An example of such a diagnostic tree is shown in Figure 5. In this example, a motorist is contemplating some of the possible causes of his car's failure to start.

The tree could be extended to take account of more and more detail, but as it stands, there are nine terminal nodes,

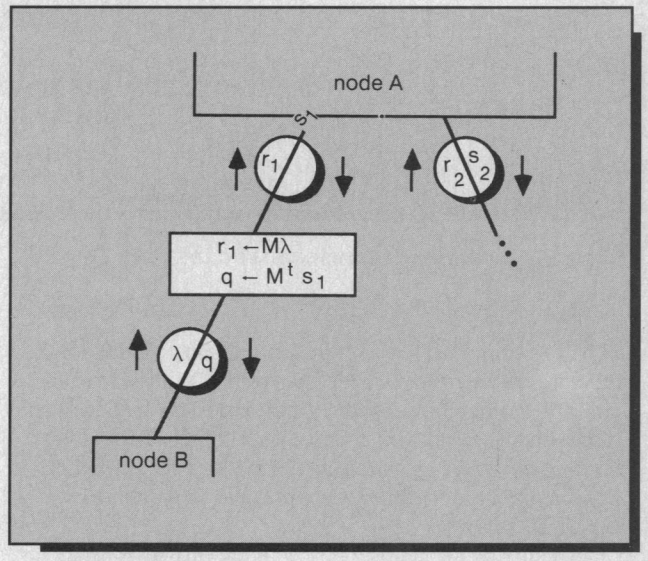


Figure 4. A typical link processor.

corresponding to nine mutually exclusive and exhaustive hypotheses. (We assume there is only one problem that is keeping the car from starting.) Let this set of nine hypotheses be denoted by Θ;

Θ = {fuel system at fault, other system at fault, battery weak, battery connections faulty, transmission not in park or neutral, some other aspect of starting system at fault, ignition switch defective, starter relay defective, some other switch defective}

The set Θ is called the *frame of discernment* for the problem. Notice that each node in the tree corresponds to a subset of Θ. The terminal nodes correspond to singleton subsets; for example,

G = {battery weak }

The other subsets correspond to larger subsets; for example,

C = {battery weak, battery connections faulty }

The topmost node corresponds to the whole set Θ.

Gordon and Shortliffe suggest that each item of diagnostic evidence will bear most directly on a single node of the tree, either supporting or refuting the hypothesis represented by that node. The problem is to combine all this diagnostic evidence. As Gordon and Shortliffe point out, this problem

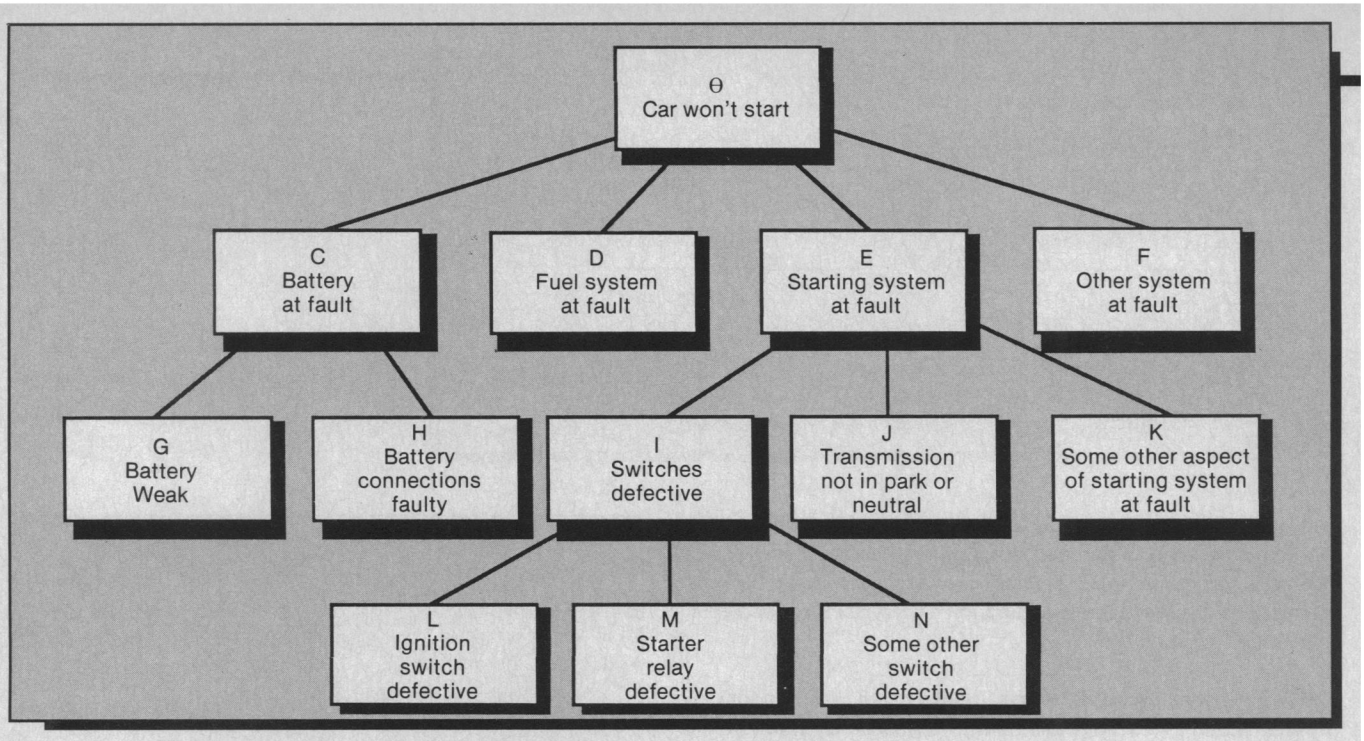


Figure 5. An example of a diagnostic tree.

can be formulated in terms of belief functions. In this formulation, each item of evidence is represented by a special kind of belief function called a simple support function. Each simple support function is focused on a subset of Θ corresponding to a node or on the complement of such a subset. These simple support functions must then be combined by Dempster's rule.

The reader who is not familiar with the theory of belief functions will need to consult Shafer.[6] We can, however, provide some insight into the mathematics of the theory by relating it to the idea of a random subset. A function Bel which assigns a number to every subset A of a finite frame Θ is a *belief function* if there exists a random non-empty subset S of Θ such that $\text{Bel}(A) = Pr(S \subseteq A)$. A subset A of Θ is a *focal element* of Bel if there is a non-zero probability that S will equal A. The belief function Bel is a *simple support function* focused on A if A and Θ are its only focal elements. It is *vacuous* if Θ is its only focal element. *Dempster's rule of combination* is a rule for calculating a new belief function from two or more belief functions. The result of combining Bel_1 and Bel_2 by this rule is denoted by $\text{Bel}_1 \oplus \text{Bel}_2$ and is called the orthogonal sum of Bel_1 and Bel_2. Intuitively, $\text{Bel}_1 \oplus \text{Bel}_2$ represents the result of pooling the evidence represented by the separate belief functions whenever these items of evidence are independent. Mathematically, $\text{Bel}_1 \oplus \text{Bel}_2$ is the belief function corresponding to a random subset that has the distribution of $S_1 \cap S_2$ conditional on $S_1 \cap S_2$ being non-empty, under the assumption that S_1 and S_2 are independent and correspond to Bel_1 and Bel_2, respectively.

Dempster's rule can involve a prohibitive amount of computation when the frame Θ is large. Shafer and Logan[1] showed, however, that efficient implementation of the rule is possible in the problem formulated by Gordon and Shortliffe. The efficiency of their method derives mainly from the fact that it does not actually perform combinations on the whole frame Θ. Instead it operates on local families of hypotheses. Like Pearl's scheme, Shafer and Logan's scheme can be implemented so that its computations are both locally carried out and locally controlled. Here, however, we will follow Shafer and Logan's original exposition, in which the computations are globally controlled. The global control calls for computations moving up the tree and then back down.

Let ϑ denote the collection of all non-initial nodes. Let us call a set of nodes that consists of all the daughters of a given non-terminal node a *sib*. Let S_A denote the sib consisting of the daughters of A. A non-terminal node together with its sib is a *family*. We suppose that for each node A in ϑ, we have one simple support function focused on A and another focused on the complement $\bar{A}$. The goal is to combine these belief functions by Dempster's rule to get overall degrees of belief for and against each node.

We begin by combining the two simple support functions for each A in ϑ. This yields a single dichotomous belief function Bel_A with dichotomy $\{A, \bar{A}\}$. We assume that $\text{Bel}_A(A)$ and $\text{Bel}_A(\bar{A})$ are both strictly less than one, but we allow either or both to be zero. For any node A in the tree, we denote by $\text{Bel}_A{\downarrow}$ the orthogonal sum of all Bel_B for all nodes B in ϑ that are strictly below A. In Figure 5, for example, $\text{Bel}_C{\downarrow} = \text{Bel}_G \oplus \text{Bel}_H$, and $\text{Bel}_\Theta{\downarrow} = \text{Bel}_C \oplus \text{Bel}_D \oplus \text{Bel}_E \oplus \text{Bel}_F \oplus \text{Bel}_C{\downarrow} \oplus \text{Bel}_E{\downarrow}$. If A is a terminal node then $\text{Bel}_A{\downarrow}$ is vacuous. For each node A, we denote by $\text{Bel}_A{\uparrow}$ the orthogonal sum of Bel_B for all node B in ϑ that are neither below A nor equal to A. $\text{Bel}_\Theta{\uparrow}$ is vacuous. For any node A, we set $\text{Bel}_A{}^L = \text{Bel}_A \oplus \text{Bel}_A{\downarrow}$ and set $\text{Bel}_A{}^U = \text{Bel}_A \oplus \text{Bel}_A{\uparrow}$. Our goal, of course, is to calculate values of $\text{Bel}^T = \oplus \{\text{Bel}_A \mid A \in \vartheta\}$. Note that $\text{Bel}^T = \text{Bel}_A \oplus \text{Bel}_A{\uparrow} \oplus \text{Bel}_A{\downarrow} = \text{Bel}_A{}^L \oplus \text{Bel}_A{\uparrow} = \text{Bel}_A{}^U \oplus \text{Bel}_A{\downarrow}$ for any node A.

The algorithm can be divided into three stages—Stage 1 goes up the tree, Stage 2 goes down the tree, and Stage 3 computes total beliefs for all nodes in ϑ. We sketch each stage here and give details in the appendix.

Stage 1 (*Going up the tree*). Let $\{A\} \cup S_A$ be a family for which the following values are available: $\text{Bel}_B{}^L(B)$ and $\text{Bel}_B{}^L(\bar{B})$ for every $B \in S_A$. First compute

$$\text{Bel}_A{\downarrow}(A) = \oplus\{\text{Bel}_B{}^L \mid B \in S_A\}(A)$$

and $\text{Bel}_A{\downarrow}(\bar{A})$ likewise. Then compute

$$\text{Bel}_A{}^L(A) = (\text{Bel}_A \oplus \text{Bel}_A{\downarrow})(A)$$

and $\text{Bel}_A{}^L(\bar{A})$ likewise. These computations can be performed using $\{\bar{A}\} \cup S_A$ as the frame, and they will therefore be feasible if A does not have too many daughters. Additional efficiency can be gained by using Barnett's[7] technique for the first calculation.

We begin with families whose sibs are terminal nodes (since if B is a terminal node, then $\text{Bel}_B{}^L = \text{Bel}_B$), combine the belief functions attached to them to find degrees of belief for and against their mothers, then do the same for the mother's mother, and so on until we have $\text{Bel}_A{}^L(A)$ and $\text{Bel}_A{}^L(\bar{A})$ for every node $A \in \vartheta$. It is not necessary to apply this stage to the family whose mother is the initial node Θ.

Stage 2 (*Going down the tree*). Let $\{A\} \cup S_A$ be a family for which the following information is available: $\text{Bel}_B{}^L(B)$ and $\text{Bel}_B{}^L(\bar{B})$ for all $B \in S_A$, $\text{Bel}_A{}^U(A)$, and $\text{Bel}_A{}^U(\bar{A})$. First, for each B in S_A we compute

$$\text{Bel}{\downarrow}(B) = (\text{Bel}_A{}^U \oplus (\oplus\{\text{Bel}_C{}^L \mid C \in S_A \setminus \{B\}\}))(B)$$

and $\text{Bel}{\downarrow}(\bar{B})$ likewise. Then for each $B \in S_A$ we compute

$$\text{Bel}_B{}^U(B) = (\text{Bel}_B \oplus \text{Bel}_B{\uparrow})(B),$$

and $\text{Bel}_B{}^U(\bar{B})$ likewise. Again, the computations use $\{\bar{A}\} \cup S_A$ as their frame and the first step can use Barnett's technique.

We begin Stage 2 for the family whose mother is the initial node Θ, since it is the only family that has the required values ($\text{Bel}_\Theta{}^U(\Theta) = 1$ and $\text{Bel}_\Theta{}^U(\bar{\Theta}) = 0$). We then continue Stage 2 for all those families whose mothers are daughters of Θ, and so on until we have $\text{Bel}{\uparrow}(A)$ and $\text{Bel}{\uparrow}(\bar{A})$ for all $A \in \vartheta$. It is not necessary to perform Step 2 for terminal nodes.

Stage 3 (*Computing total beliefs*). For each node A, we compute

$$\text{Bel}^T(A) = (\text{Bel}_A{\uparrow} \oplus \text{Bel}_A{}^L)(A)$$

and $\text{Bel}^T(\bar{A})$ likewise. Here we use $\{A,\bar{A}\}$ as our frame.

The scheme assumes that at each node A, we store the following twelve belief values: $\text{Bel}_A(A)$, $\text{Bel}_A(\bar{A})$, $\text{Bel}_A{\downarrow}(A)$, $\text{Bel}_A{\downarrow}(\bar{A})$, $\text{Bel}_A{}^L(A)$, $\text{Bel}_A{}^L(\bar{A})$, $\text{Bel}_A{\uparrow}(A)$, $\text{Bel}_A{\uparrow}(\bar{A})$, $\text{Bel}_A{}^U(A)$, $\text{Bel}_A{}^U(\bar{A})$, $\text{Bel}^T(A)$, and $\text{Bel}^T(\bar{A})$. Storage of all these values will minimize computation. If space is more at a premium than processor time, then it is sufficient to store just six of these values (the values for A and $\bar{A}$ of Bel_A, $\text{Bel}_A{\uparrow}$, and $\text{Bel}_A{\downarrow}$) and compute the rest as needed.

Qualitative Markov trees

The concept of conditional independence is familiar from probability theory, and it leads within probability theory to many other concepts, including Markov chains and Markov networks. In this section, we introduce a purely qualitative (non-probabilistic) concept of conditional independence and the corresponding concept of a qualitative Markov tree. Qualitative Markov trees are the setting for our general computational scheme for propagating belief functions.

Recall that a set Φ of subsets of Θ is a *partition* of Θ if the sets in Φ are all non-empty and disjoint, and their union is Θ. Both belief functions and random variables can be described qualitatively by partitions. Given a random variable X defined on Θ, the partition Φ of Θ *associated* with X is defined as follows: two distinct outcomes in Θ belong to the same set P in Φ if the real numbers assumed by X when each of these two outcomes happen are the same. Given a belief function Bel on a frame Θ, the partition *associated* with Bel is the coarsest partition of Θ that carries Bel. (A partition *carries* Bel if the focal elements of Bel are all unions of elements of Φ.)

A partition Φ of a frame Θ can itself be regarded as a frame. If Bel is a belief function on Θ, then the *coarsening* of Bel to Φ is the belief function Bel_Φ on Φ given by $\text{Bel}_\Phi(\{P_1,\ldots,P_k\}) = \text{Bel}(P_1 \cup \ldots \cup P_k)$ for every subset $\{P_1,\ldots,P_k\}$ of Φ. If Bel is a belief function on Φ, then the *vacuous extension* of Bel to Θ is the belief function Bel^Θ given by $\text{Bel}^\Theta(A) = \text{Bel}(\cup\{P \mid P \subseteq A, P \in \Phi\})$. If a belief function is carried by Φ, then Bel_Φ contains all the information about Bel. In fact, in this case, Bel can be recovered from Bel_Φ by vacuous extension: $(\text{Bel}_\Phi)^\Theta = \text{Bel}$. If Φ_1 and Φ_2 are two partitions, and Bel is a belief function on Φ_1, then the *projection* of Bel to Φ_2 is the result of vacuously extending Bel to Θ and then coarsening to Φ_2.

Recall that the *coarsest common refinement* of $\Phi_1, \ldots, \Phi_n$, written as $\Phi_1 \wedge \ldots \wedge \Phi_n$ or as $\wedge\{\Phi_j \mid j = 1, \ldots, n\}$, is the partition

$$\{P_1 \cap \ldots \cap P_n \mid P_j \in \Phi_j \text{ for all } j, \text{ and } P_1 \cap \ldots \cap P_n \neq \phi\}.$$

We say that the partitions $\Phi_1, \ldots, \Phi_n$ are *qualitatively conditionally independent* given the partition Φ, written as $[\Phi_1,\ldots,\Phi_n] \perp \Phi$, if $P \cap P_1 \cap \ldots \cap P_n \neq \phi$ whenever $P \in \Phi$, $P_i \in \Phi_i$ for all i, and $P \cap P_i \neq \phi$ for all i. This definition does not involve probabilities, just logical relations. But stochastic conditional independence does imply qualitative conditional independence.

Lemma 1. Let $X, X_1, \ldots, X_n$ be random variables defined on a finite sample space Θ, and let $Pr: 2^\Theta \to [0, 1]$ be a probability distribution on Θ such that $Pr(\Theta) > 0$ for all $\Theta \in \Theta$. If $X_1, \ldots, X_n$ are conditionally independent given X, then $\Phi_1, \ldots, \Phi_n$ are qualitatively conditionally independent given Φ, where Φ is the partition associated with X, and Φ_i is the partition associated with X_i for $i = 1, \ldots, n$.

Qualitative conditional independence is important for belief functions because it is used in defining the circumstances under which we get the right answer when we implement Dempster's rule on a partition rather than on a finer frame. This is expressed technically by saying that if Bel_1 and Bel_2 are carried by Φ_1 and Φ_2, respectively, and $[\Phi_1, \Phi_2] \perp \Phi$, then $(\text{Bel}_1 \oplus \text{Bel}_2)_\Phi = (\text{Bel}_1)_\Phi \oplus (\text{Bel}_2)_\Phi$.[8]

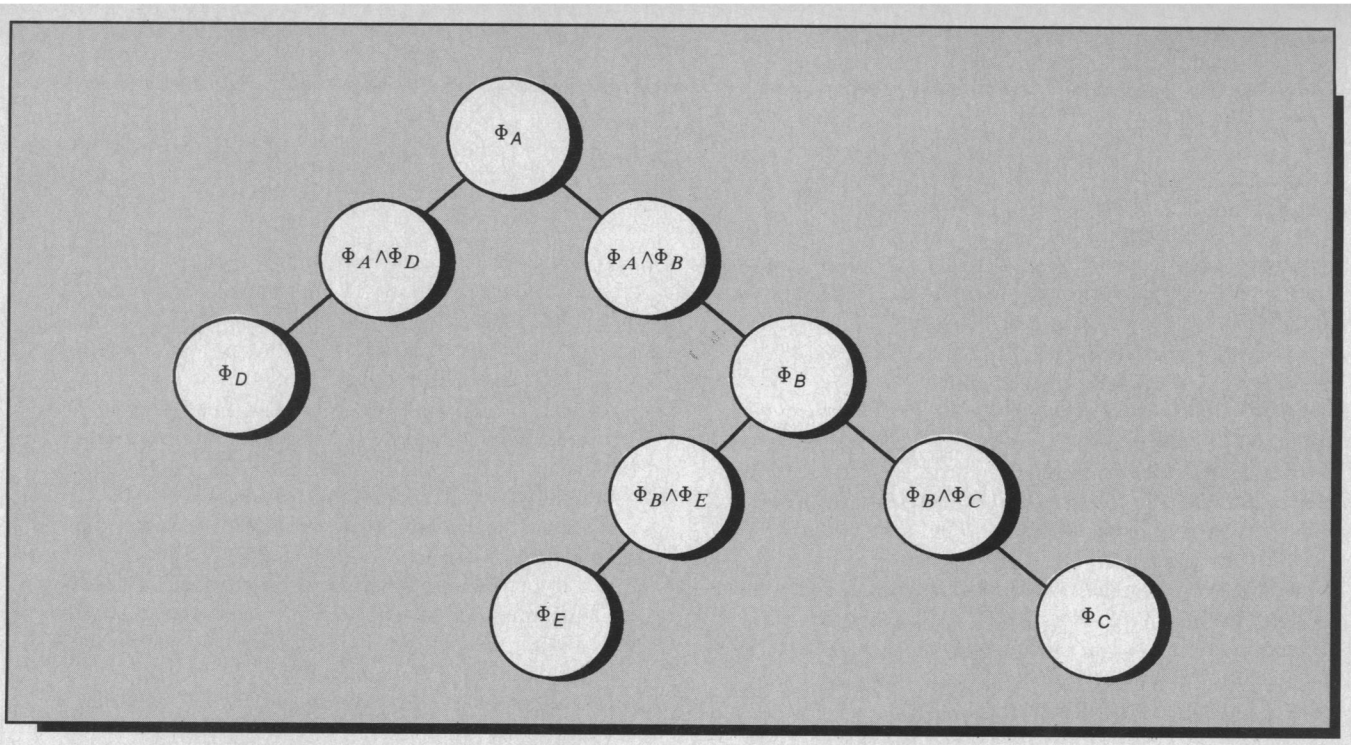


Figure 6. A qualitative Markov tree constructed from the causal tree in Figure 1.

An (undirected) *network* is a pair (J, E), where J, the *nodes* of the network is a finite set, and E, the *links* of the network is a set of unordered pairs of distinct elements of J. We say the elements i and j of J are *adjacent* or *neighbors* if $(i, j) \in E$. A network is a *tree* if it is connected and there are no cycles.

Let $T = (J, E)$ be a tree. Given any node i in J, deletion of i from j and deletion of all edges incident to i from E results in a forest of k subtrees. Let the collection of nodes in the jth subtree be denoted by $\alpha_j(i)$.

Definition. Let $\{\Phi_j \mid j \in J\}$ be a finite collection of partitions and let $T = (J, E)$ be a tree. We say that $\{\Phi_j \mid j \in J\}$ is *qualitative Markov* with respect to T or equivalently that T is *qualitative Markov* for $\{\Phi_j \mid j \in J\}$, if for every i in J, the coarsest common refinements of partitions in $\alpha_m(i)$ for $m = 1, \ldots, k$ are qualitatively conditionally independent given the partition Φ_i, i.e.,

$$[\wedge \{\Phi_j \mid j \in \alpha_1(i)\}, \ldots, \wedge \{\Phi_j \mid j \in \alpha_k(i)\}] \dashv \Phi_i.$$

Using Lemma 1, we can show that a Bayesian causal tree becomes a qualitative Markov tree when we associate with each node B the partition Φ_B associated with the random variable corresponding to B. It remains a qualitative Markov tree if we interpolate between each daughter node and its mother a node corresponding to the common refinement of the two. Figure 6 shows the result of this interpolation for the causal tree of Figure 1.

A qualitative Markov tree can also be constructed from a diagnostic tree. Indeed, if $\{A\} \cup S_A$ is a family in the diagnostic tree, then $\{\overline{A}\} \cup S_A$ is a partition of Θ. The qualitative Markov tree has a node for each of these partitions, and it has a link between $\{\overline{A}\} \cup S_A$ and $\{\overline{B}\} \cup S_B$ whenever B is a daughter of A. Figure 7 shows the qualitative Markov tree that is constructed in this way from Figure 5. (Notice that the topmost family consists just of the daughters of Θ; $\overline{\Theta}$ is not included since it is the empty set.)

A qualitative Markov tree identified in this way remains a qualitative Markov tree if $\{A, \overline{A}\}$ is interpolated between $\{\overline{A}\} \cup S_A$ and the family in which A is a daughter. We may also link $\{A, \overline{A}\}$ to A's family if A is a terminal node. Figure 8 shows the qualitative Markov tree that results when such dichotomies are added to Figure 7.

Propagating belief functions in qualitative Markov trees

Our general computational scheme applies to any qualitative Markov tree. Each of the belief functions being combined must be carried by one of the partitions in the tree. The efficiency of the scheme will depend on the size of the partitions, since Dempster's rule increases in complexity exponentially with the size of the frame[7] and the essence of the

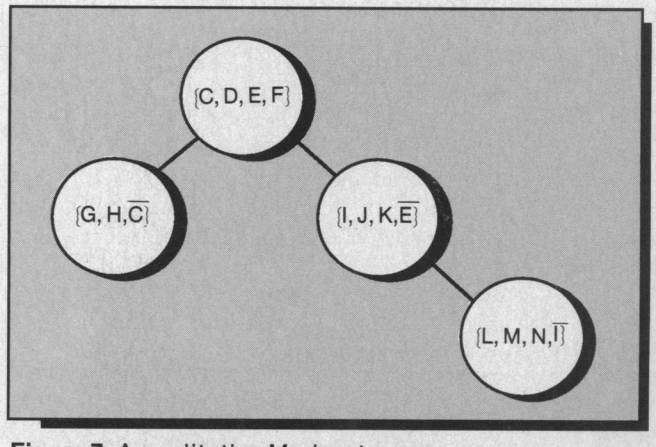


Figure 7. A qualitative Markov tree constructed from the diagnostic tree in Figure 5.

scheme is to substitute multiple implementations of the rule over each partition for a single implementation over the whole frame Θ.

We imagine that a processor is located at each node. The processors run in parallel, without synchronization. There is direct communication only between processors at adjacent nodes. The processor at node Φ_i combines belief functions using Φ_i as a frame and also projects belief functions from Φ_i to its neighbors.

The belief functions we want to combine are input directly to the nodes; each belief function is input to a node corresponding to a partition that carries it. The eventual result is that the processor at each node arrives at the coarsening to that node of the orthogonal sum of all the belief functions that have been input.

It does not matter when belief functions are input to the processors; additional belief functions may be input after the system has reached equilibrium. For the sake of exposition, however, let us suppose that all belief functions are input at the beginning, and that the processor at Φ_i initially combines the belief functions input to it to form a belief function Bel_i. In this case, the final result at Φ_i should be $(\oplus\{\text{Bel}_i \mid i \in J\})\Phi_i$, the combination of all the Bel_i coarsened to Φ_i.

Figure 9 shows a typical processor. As this figure shows, each node shares random access memory with each of its neighbors. The memory shared between two adjacent nodes, say i and j, is divided into two parts: node i has read-only access (ROA) to one part and write-only access (WOA) to the other part. The read-only access part for node i is where node j writes messages it transmits to node i. The write-only access part for node i is where node j has read-only access and is where node i writes information it wishes to transmit to node j.

The belief functions calculated by the processor at i are denoted by $(\text{Bel}^T)\Phi_i$ and $\text{Bel}_{i,y}$. $(\text{Bel}^T)\Phi_i$ is the belief function for Φ_i calculated on the total evidence received by i so far. $\text{Bel}_{i,y}$ is the belief function most recently transmitted from i to y.

Let $N_T(i)$ represent the set of all neighbors of node i in the tree T. Initially, Bel_i is entered at each node in the tree. The processor at i then projects Bel_i to each of its neighbors; for each $X \in N_T(i)$, it writes $\text{Bel}_{i,x} \leftarrow (\text{Bel}_i)\Phi_x$ in the WOA portion of the memory it shares with x. At constant or random intervals, each processor scans all ROA sections of its memory. If i reads something new, it (1) computes $(\text{Bel}^T)\Phi_i \leftarrow \oplus\{\text{Bel}_{x,i} \mid x \in N_T(i)\} \oplus \text{Bel}_i$ and stores this information in its cumulative belief function registers and (2) computes $\text{Bel}_{i,y} \leftarrow (\oplus\{\text{Bel}_{x,i} \mid x \in N_T(i) \setminus \{y\}\} \oplus \text{Bel}_i)\Phi_y$ for each $y \in N_T(i)$ and writes this information in the WOA portion of the memory it shares with node y. Part 2 of the computation involves projection as well as combination by Dempster's rule. The details of the computation involved in this projection will depend, of course, on the details of the tree.

This process is repeated until nothing new is read by any node in any of the ROA sections of the memory it shares with its neighbors. When this condition is reached, we say the network is in equilibrium. It can be shown that at equilibrium, the

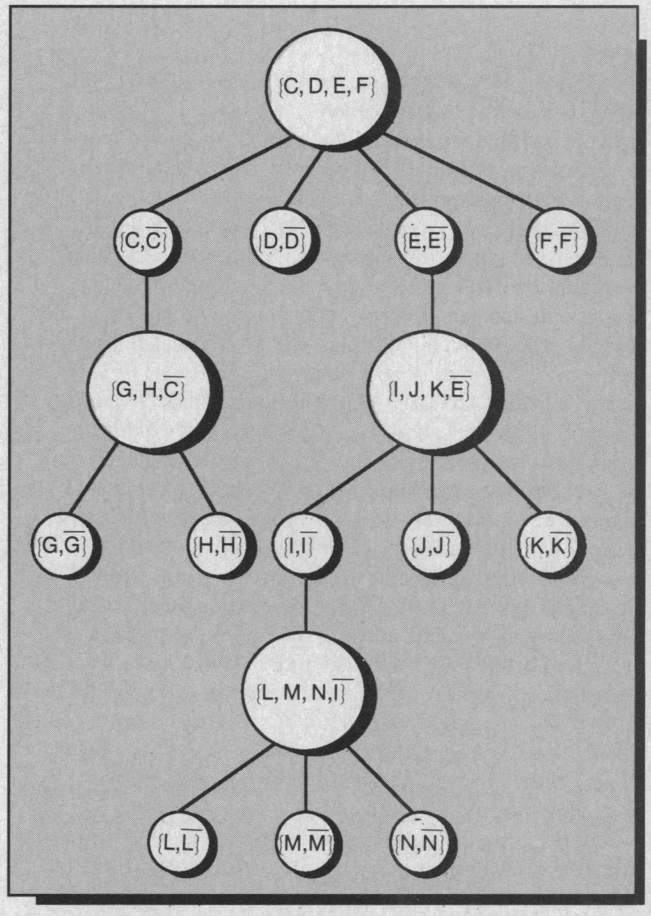


Figure 8. The enlarged qualitative Markov tree for the car that won't start.

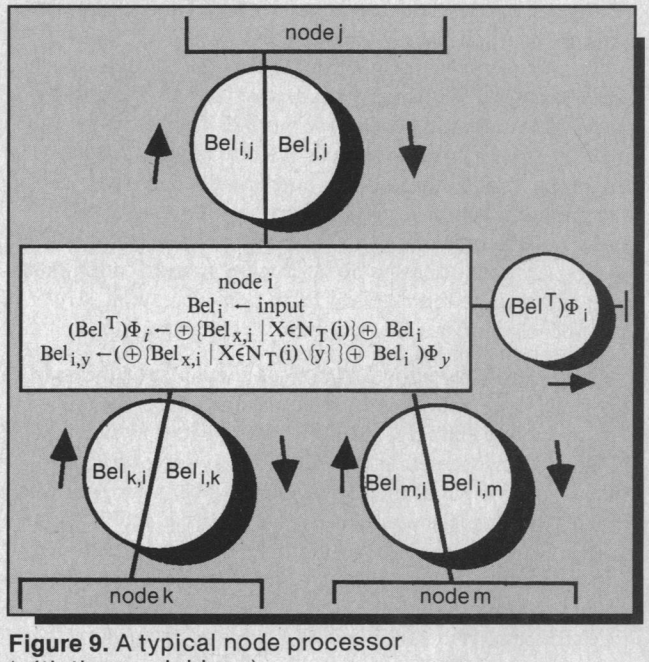


Figure 9. A typical node processor (with three neighbors).

belief function in the cumulative belief function register of node i will be

$$\oplus\{\mathrm{Bel}_{x,i} \mid x \in N_T(i)\} \oplus \mathrm{Bel}_i = (\oplus\{\mathrm{Bel}_j \mid j \in J\})\Phi_i$$

which represents the total belief coarsened to the partition at node i. The time required to reach equilibrium is proportional to the maximum diameter of the tree (the length of the longest path between two nodes in the tree).

We have already asserted that Pearl's computational scheme for Bayesian causal trees and Shafer and Logan's computational scheme for diagnostic trees are special cases of this general scheme. This is demonstrated in detail by Shafer, Shenoy, and Mellouli.[9] Here we will merely sketch how the two schemes fit in as special cases.

To fit Pearl's scheme into our general scheme, we shift from the Bayesian causal tree of Figure 1 to the qualitative Markov tree of Figure 6. There are processors for each node in this qualitative Markov tree; the processor at node A in the causal tree is associated with Φ_A in the qualitative Markov tree, and the processor at the link between A and B is associated with $\Phi_A \wedge \Phi_B$ in the qualitative Markov tree. The initial inputs into the Bayesian causal tree consisted of a prior probability distribution at the initial node and matrices of transition probabilities between nodes. The prior probability distribution is a belief function. The matrix M of transition probabilities from A to B is not a belief function but it can be represented by a belief function. What is needed is a belief function Bel_M on $\Phi_A \wedge \Phi_B$ such that $(\mathrm{Bel}_M)\Phi_A$ and $(\mathrm{Bel}_M)\Phi_B$ are vacuous but $(\mathrm{Bel}_M \oplus \mathrm{Bel}_{a_i})(\{b_j\}) = M_{ij}$, where Bel_{a_i} is a belief function that has $A = \{a_i\}$ as its only focal element. There are many such belief functions Bel_M, and the choice among them does not matter; see Shafer.[10] Finally, the new evidence taken into account by Pearl's scheme consists of observed values for terminal nodes. Each of these observed values can be represented by a belief function that has it as its only focal element.

Once we see Pearl's scheme as a special case of belief function scheme, we see that any causal tree used by Pearl's Bayesian analysis can also be used for more flexible belief-function analyses. We might, for example, want to relate a daughter and mother node using a belief function that falls short of determining a complete probability transition matrix. We might want to represent evidence bearing on the initial node by a belief function that is not a probability distribution, and we might want to represent evidence bearing on other nodes using belief functions that are more complex than likelihoods.

To see that Shafer and Logan's scheme is a special case of our general scheme, we shift from the diagnostic tree of Figure 5 to the enlarged qualitative Markov tree shown in Figure 8. The evidence in the form of dichotomous belief functions are input at the corresponding dichotomous nodes in Figure 8. There are no non-vacuous belief functions input at the nodes corresponding to families in the tree. The first computation in Stage 1 of Shafer and Logan's scheme consists of a family node combining all the belief functions received from its dichotomous daughters and projecting the sum to its dichotomous mother. The second computation in this stage consists of combining this projection and the belief function input at the dichotomous mother node. The first computation in Stage 2 consists of a family node projecting to each of its dichotomous daughters the sum of the belief functions received from all other daughters and the mother. The second computation in this stage consists of each dichotomous daughter of the family node combining the belief function input to it and the belief function it receives from its mother. Stage 3 of the scheme simply consists of each dichotomous node combining the belief functions it receives from its mother, daughter and the belief function input to it from outside. The timing of the stages in Shafer and Logan's scheme simply serves to minimize computations assuming one central serial processor. If minimizing the total number of computations is not important (as will be the case if there are several processors running in parallel), then of course, the timing aspect of the Shafer and Logan scheme is not important. This is because in the general scheme, each processor does the computations each time it receives a new message (projection) from one or more of its neighbors. Hence, only the most recent messages received from the neighbors matter in the final result.

Once we see that the Shafer and Logan's scheme is a special case of our scheme, we see that the qualitative Markov tree resulting from a diagnostic tree can be used for more flexible analysis. For example, we could input at each family node a more complicated belief function that is carried by the partition represented by the family node. This is the generalization of the Shafer and Logan's scheme described in Shafer.[4] In this case, Barnett's technique can no longer be applied but the computations remain feasible if the families are not too large.

In this article we have described a computational scheme that is applicable whenever evidence can be broken down into independent components that bear most directly on the nodes of a qualitative Markov tree. We believe that this scheme is promising for use in expert systems, but fulfillment of the promise will require methods for helping users of expert systems construct qualitative Markov trees that fit the problem and their evidence.

Pearl[5] has proposed methods for constructing Bayesian causal trees from joint probability distributions. These methods are intriguing, but their use is clearly limited to the case where quite detailed meaningful joint distributions are available. Straightforward interactive systems for constructing diagnostic trees are possible, but this also will have limited applicability.

We believe that broad use of our scheme will require interactive systems that take advantage of people's general ability to decompose evidence and their ability to make judgments of qualitative independence. What is needed are systems that can use such judgments to construct qualitative Markov trees. ■

Appendix

Here we describe in detail the computations involved in Stages 1 to 3 of the scheme for diagnostic trees. The details of Stage 2 and Stage 3 computations are slightly different from those described in Shafer and Logan.[1]

Let us use the following notation:

$\text{Bel}_A(A) := A^o$ $\quad \text{Bel}_A{}^L(A) := A^L$ $\quad \text{Bel}_A{}^U(A) := A^U$

$\text{Bel}_A(\bar{A}) := \bar{A}^o$ $\quad \text{Bel}_A{}^L(\bar{A}) := \bar{A}^L$ $\quad \text{Bel}_A{}^U(\bar{A}) := \bar{A}^U$

$\text{Bel}_A{\downarrow}(A) := \text{A}{\downarrow}$ $\quad \text{Bel}_A{\uparrow}(A) := A{\uparrow};$ $\quad \text{Bel}^T(A) := A^T$

$\text{Bel}_A{\downarrow}(\bar{A}) := \bar{\text{A}}{\downarrow}$ $\quad \text{Bel}_A{\uparrow}(\bar{A}) := \bar{\text{A}}{\uparrow}$ $\quad \text{Bel}^T(\bar{A}) := \bar{A}^T.$

Stage 1 (Going up the tree).

Step 1. Calculating $A{\downarrow}$ and $\bar{A}{\downarrow}$ from B^L and $\bar{B}^L$ for B in S_A :

$$A{\downarrow} = 1 - K,$$

$$\bar{A}{\downarrow} = K\,(\Pi\,\{\bar{B}^L{\downarrow}\,(1 - B^L) \mid B \in S_A\}), \text{ where}$$

$$K = 1 / (1 + \Sigma\,\{B^L \mid (1 - B^L) \mid B \in S_A\}).$$

Step 2. Calculating A^L and $\bar{A}^L$ from A^o, $\bar{A}^o$, $A{\downarrow}$, and $\bar{A}{\downarrow}$:

$$A^L = 1 - K(1 - A^o)(1 - A{\downarrow}),$$

$$\bar{A}^L = 1 - K(1 - \bar{A}^o)(1 - \bar{A}{\downarrow}), \text{ where}$$

$$K = 1 / (1 - A^o\bar{A}{\downarrow} - \bar{A}^o\,A{\downarrow}).$$

Stage 2 (Going down the tree).

Step 1. Calculating $B{\uparrow}$ and $\bar{B}{\uparrow}$ for B in S_A from A^U, $\bar{A}^U$ and C^L, $\bar{C}^L$ for C in $S_A \setminus \{B\}$:

$$B{\uparrow} = 1 - K,$$

$$\bar{B}{\uparrow} = K\,(\Pi\,\{\bar{C}^L / (1 - C^L) \mid C \in S_A \setminus \{B\}\})\,(A^U / (1 - \bar{A}^U)),$$

where

$$K = 1 / (1 + \Sigma\,\{C^L / (1 - C^L) \mid C \in S_A \setminus \{B\}\} + (\bar{A}^U / (1 - \bar{A}^U))).$$

Step 2. Calculating B^U and $\bar{B}^U$ from B^o, $\bar{B}^o$, $B{\uparrow}$, and $\bar{B}{\uparrow}$:

$$B^U = 1 - K(1 - B^o)\,(1 - B{\uparrow}),$$

$$\bar{B}^U = 1 - K(1 - \bar{B}^o)\,(1 - \bar{B}{\uparrow}), \text{ where}$$

$$K = 1 / (1 - B^o\bar{B}{\uparrow} - \bar{B}^o\,B{\uparrow}).$$

Stage 3 (Calculating total belief for a node).

Calculating A^T and $\bar{A}^T$ for any A in ϑ from $A{\uparrow}$,$\bar{A}{\uparrow}$, A^L, and $\bar{A}^L$:

$$A^T = 1 - K(1 - A{\uparrow})\,(1 - A^L),$$

$$\bar{A}^T = 1 - K(1 - \bar{A}{\uparrow})\,(1 - \bar{A}^L), \text{ where}$$

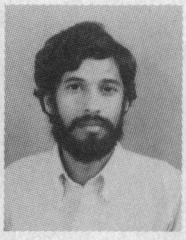
$$K = 1 / (1 - A{\uparrow}\bar{A}^L - \bar{A}{\uparrow}\,A^L).$$

Acknowledgment

Research for this article has been partially supported by NSF grant IST-8405210 and grant P.O. S-25780 from Harvard University.

References

1. G. Shafer and R. Logan, "Implementing Dempster's Rule For Hierarchical Evidence," working paper No. 174, 1985, School of Business, University of Kansas.
2. J. Gordon and E. H. Shortliffe, "A Method For Managing Evidential Reasoning In Hierarchical Hypothesis Spaces," *Artificial Intelligence,* Vol. 26, 1985, pp. 323-358.
3. J. Gordon and E. H. Shortliffe, "The Dempster-Shafer Theory of Evidence," *Rule-Based Expert Systems: The MYCIN Experiments of The Stanford Heuristic Programming Project,* eds., B. G. Buchanan and E. H. Shortliffe, Addison-Wesley, 1985.
4. G. Shafer, "Hierarchical Evidence," *The Second Conference on Artificial Intelligence Applications - The Engineering of Knowledge-Based Systems,* IEEE Computer Society Press, 1985, pp. 16-25.
5. J. Pearl, "Fusion, Propagation, And Structuring In Bayesian Networks," tech. report No. CSD-850022, 1985, Computer Science Department, University of California, Los Angeles.
6. G. Shafer, *A Mathematical Theory of Evidence,* Princeton University Press, 1976.
7. J. A. Barnett, "Computational Methods For A Mathematical Theory Of Evidence," *Proc. 7th Int'l Joint Conf. AI,* 1981, pp. 868-875.
8. G. Shafer, *A Mathematical Theory of Evidence,* p. 177.
9. G. Shafer, P. P. Shenoy, and K. Mellouli, "Propagating Belief Functions in Qualitative Markov Trees," working paper (in preparation), 1986, School of Business, University of Kansas.
10. G. Shafer, "Belief Functions and Parametric Models," *The Journal of the Royal Statistical Society,* Series B, Vol. 44, 1982, pp. 322-352.



Prakash P. Shenoy is an associate professor in the School of Business at the University of Kansas at Lawrence where he has been since 1978. Before that, he was with the Mathematics Research Center at the University of Wisconsin at Madison for one year.

His research interests are in the areas of artificial intelligence, information and knowledge-based systems and operations research. He has published many articles on the mathematical theory of games. He is a member of AAAI, ACM, and IEEE Computer Society.

Shenoy received a BTech in mechanical engineering from the Indian Institute of Technology at Bombay in 1973, and a MS and a PhD in operations research from Cornell University in 1975 and 1977 respectively.

Glenn Shafer is a professor in the School of Business at the University of Kansas. His current interests range from the history of probability (he is collaborating with A. W. F. Edwards and Edith Sylla on an edition of James Bernoulli's work on probability) to the applications of probability in artificial intelligence, especially expert systems and associative memory.

Shafer received an AB degree in mathematics from Princeton University in 1968. After serving in the Peace Corps in Afghanistan and attending graduate school in statistics at Berkeley and Harvard, he completed a PhD in statistics at Princeton in 1973. He taught for three years in the Statistics Department at Princeton. He has been an NSF postdoctoral fellow and a Guggenheim fellow.