Chapter 4

BELIEF UPDATING BY NETWORK PROPAGATION

Oh, would that my mind could let fall its dead ideas,
as the tree does its withered leaves.
— André Gide

This chapter deals with fusing and propagating the impact of new evidence and beliefs through Bayesian networks so that each proposition eventually will be assigned a certainty measure consistent with the axioms of probability theory. We start in Section 4.1 by arguing that any viable model of human reasoning should be able to perform this task with a self-activated propagation mechanism, i.e., with an array of simple and autonomous processors, communicating locally via the links provided by the network itself. The impact of each new piece of evidence is viewed as a perturbation that propagates through the network via message-passing between neighboring variables, with minimal external supervision. In Section 4.2 we show that these objectives can be fully realized with *causal trees,* i.e., Bayesian networks in which each node has at most one parent. In Section 4.3 we extend the result to *causal polytrees,* i.e., trees with arbitrary arrow orientation, and thus allow multiple roots and multiple parents. In both cases, we identify belief parameters, communication messages, and updating rules to guarantee that equilibrium will be reached in time proportional to the longest path in the network and that at equilibrium each proposition will be accorded a belief measure equal to its posterior probability, given all the available evidence. In Section 4.3.2, we illustrate this propagation method's evidence-pooling and credit-assignment policies with a canonical model, where multiple causes are assumed to interact disjunctively. Several approaches to achieving autonomous propagation in multiply connected networks are discussed in Section 4.4, including clustering, conditioning, and stochastic simulation. Finally, Section 4.5 extends the inferential repertoire of Bayesian networks to include answering Boolean queries under propositional constraints, with a special emphasis on conjunctive queries.

4.1 AUTONOMOUS PROPAGATION AS A COMPUTATIONAL PARADIGM

Since a fully specified Bayesian network constitutes a complete probabilistic model of the variables in a domain (i.e., it specifies a joint distribution over the variables), the network contains the information needed to answer all probabilistic queries about these variables. The queries might be requests to interpret specific input data or, if utility information is provided, requests to recommend the best course of action. Interpretation requires instantiating a set of variables corresponding to the input data, calculating their impact on the probabilities of variables designated as hypotheses, and finally, selecting the most likely combinations of these hypotheses.

In principle, once we have a joint distribution function P, the interpretation task can be performed mechanically using purely algebraic methods. For example, the impact of the observation $X_j = x_j$ on another variable X_i can be obtained from the conditional probabilities $P(x_i \mid x_j)$ for each value x_i in the domain of X_i. Using the textbook definition

$$P(x_i \mid x_j) = \frac{P(x_i, x_j)}{P(x_j)}$$

we compute $P(x_i, x_j)$ by summing the joint distribution $P(x_1, \ldots, x_n)$ over all variables except X_i and X_j. The summation can be executed in any order, but exponential savings can sometimes be realized by selecting a judicious ordering.

Network representations provide a valuable guide for making this selection. Summing over a variable, say X_k, amounts to eliminating X_k from the network while maintaining the proper dependencies among remaining variables—adding links between those neighbors of X_k that were d-separated by X_k and calculating the conditional probabilities associated with the new links. Since the size of a link matrix increases exponentially with the number of arrows that converge on a given variable, it is important to eliminate variables in an order that minimizes the number of converging arrows created throughout the process. Techniques for finding a good elimination ordering have been developed in the operations research literature [Bertelé and Brioschi 1972] and have been used for manipulating influence diagrams [Shachter 1986] (see Chapter 6). Such techniques do not, of course, avoid the exponential worst-case complexity of the interpretation task (the problem is *NP*-hard [Rosenthal 1975; Cooper 1987]), but they exploit the structural properties of sparse networks.

However, node elimination has several shortcomings. First, the process requires full supervision by a monitor that must access all parts of the network and use external computational facilities to decide, at any given stage, which variable should be eliminated next. The use of such a global supervisor is foreign to human reasoning because it requires faculties beyond most humans, e.g., comprehending

the entire structure of the network and quickly diverting attention from one section of the network to another. Our limited short-term memory, narrow range of attention, and inability to shift rapidly between alternative lines of reasoning suggest that our reasoning process is fairly local, progressing incrementally along preestablished pathways. Second, elimination techniques literally cover their tracks; they provide the final impact of a piece of evidence on a single hypothesis, but do not calculate the impact of the evidence on the nodes eliminated in the process. In many applications, we wish to know the updated belief of every variable in the network. Third, and perhaps most important, the elementary steps in the process of node elimination often have no conceptual correlates. They create and calculate spurious dependencies among variables normally perceived to be independent, and the dependencies are hard to explain in qualitative terms. Finally, elimination techniques are basically sequential, and there is growing interest in reasoning models that permit unsupervised parallelism. The interest is motivated both by technological advances in parallel computation and by the need to develop viable models of human reasoning. The speed and ease with which people perform some low-level interpretive functions, such as recognizing scenes, reading text, and even understanding stories, strongly suggest that such processes involve a significant amount of parallelism and that most of the processing is done at the knowledge level itself [Shastri and Feldman 1984].

4.1.1 Constraint Propagation and Rule-based Computation

We can model such phenomena by viewing a belief network not merely as a passive code for storing factual knowledge but also as a computational architecture for reasoning about that knowledge. This means the links in the network should be treated as the only mechanisms that direct and propel the flow of data through the process of querying and updating beliefs. Accordingly, we assume that each node in the network is given a separate processor, which maintains the parameters of belief for the host variable and manages the communication links to and from neighboring, conceptually related variables. The communication lines are assumed to be open at all times, i.e., each processor may, at any time, interrogate its neighbors and compare their parameters to its own. If the compared quantities satisfy some local constraints, no activity takes place. However, if any constraint is violated, the responsible node is activated and the violating parameter corrected. This, of course, activates similar revisions at neighboring nodes and initiates a multidirectional propagation that will continue until equilibrium is reached.

EXAMPLE 1: To illustrate the process of constraint propagation, consider the graph coloring problem depicted by Figure 4.1. Each node in the graph must be assigned one of

three colors, {1, 2, 3}, in such a way that no two adjacent nodes will have identical colors. The constraints in this problem are local; no node can assume a color seen at any of its neighbors. A local approach would be to assign a processor to each node and have it compare its current color to the colors of its neighbors. If equality is discovered, the processor should choose a new color, different (if possible) from the neighbors' colors. We assume there is no synchronization, so all possible activation schedules could be realized.

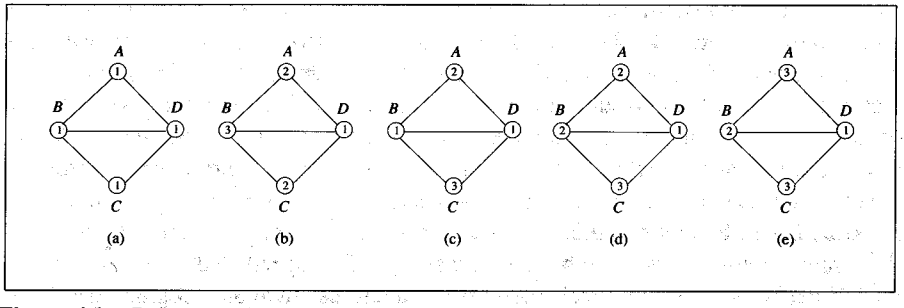

Figure 4.1. *Demonstrating constraint propagation in the graph coloring problem.*

Figure 4.1*a* shows the initial state of the system: all nodes are colored 1. Figure 4.1*b* shows the configuration after the nodes are activated in the order A, B, C, D: a stable solution is established once C selects the color 2. If, instead, the activation schedule happens to be A, C, B, D, and if C selects the color 3, a deadlock occurs (Figure 4.1*c*): B cannot find a color different from all its neighbors. A way out of such deadlock is to instruct each processor to change its color arbitrarily if no better one can be found. Indeed, if B changes its color to 2 (temporarily conflicting with A, as in Figure 4.1*d*), it forces A to choose the color 3, thus realizing a global solution (Figure 4.1*e*).

Even if a global solution exists, there is no guarantee, in general, that it will be found by repeated local relaxations. However, if the escape from deadlock is totally random, the probability of reaching a solution within a given time t approaches 1 as t increases. Note also that if the node activation is both parallel and synchronous, the system can fall into an indefinite dynamic loop without reaching a solution. For example, starting with the state of Figure 4.1*a*, all processors may simultaneously choose the color 2, then 3, etc. Such pathological behavior will not occur in the networks that we shall deal with.

One of the main reasons for adopting this distributed computation paradigm in evidential reasoning tasks is that it automatically exploits the independencies embodied in the network, via subtask decomposition, to gain a substantial reduction in complexity. For example, if the Markov neighbors of some variable X have successfully computed their combined distribution function, X can compute its own distribution without interacting with any variable outside its neighborhood (see Eq. (3.12)). Moreover, if the network has a tree structure, then X can compute its distribution by consulting each of its neighbors separately. If any of X's neighbors undergo change, X updates its own distribution and reports the update to the other neighbors, and so on until, at the network's periphery, we meet evidential variables whose probabilities are predetermined and the process halts.

The second reason to adopt this distributed paradigm is that it leads to a "transparent" revision process, in which the intermediate steps can be given an intuitively meaningful interpretation. Because a distributed process allows each computational step to obtain input only from neighboring, semantically related variables, and because the activation of these steps proceeds along semantically familiar pathways, people find it easy to give meaningful interpretation to the individual steps and thus gain confidence in the final result. It is also possible to generate qualitative justifications mechanically by tracing the sequence of operations along the activated pathways and giving them causal or diagnostic interpretations using appropriate verbal expressions.

The ability to update beliefs by an autonomous propagation mechanism also has a profound effect on sequential implementations of evidential reasoning. Of course, when this architecture is simulated on sequential machines, the notion of autonomous processors working simultaneously is only a metaphor, but it signifies the complete separation of the stored knowledge from the control mechanism. This separation is the proclaimed, if rarely achieved, goal of rule-based systems. It guarantees the utmost flexibility for a sequential controller; the computations can be performed in any order, with no need to remember or verify which parts of the network have already been updated. Thus, belief updating may be activated by changes occurring in logically related propositions (*spreading activation*), by requests for evidence arriving from a central supervisor (*goal-directed activation*), by a predetermined schedule, or entirely at random. The communication and interaction among individual processors can be simulated using a blackboard architecture [Lesser and Erman 1977], where each proposition is designated specific areas of memory to access and modify.

Finally, the uniformity of this propagation scheme makes it easy to formulate in object-oriented languages: the nodes are objects of the same generic type, and the belief parameters are the messages by which interacting objects communicate. The programmer need only specify how a single object reacts to changes occurring at its neighbors; he need not provide timing information or say where to store partial results.

Constraint propagation mechanisms have a special appeal for AI researchers because they are similar in many respects to logical inference rules. We already have seen that an inference rule of the form "If premise A, then action B" constitutes a very attractive unit of computation for three reasons:

1. The triggering mechanism is local, i.e., it grants a license to initiate the action whenever the premise A is true in the knowledge base K, regardless of the other information that K contains.
2. The action is computationally simple and normally involves only a minor adjustment in the knowledge base.
3. Both the triggering mechanism and the action are conceptually meaningful and are therefore easy to program, modify, and explain.

Almost identical features hold in the constraint propagation formalism. Each processor receives a permanent license to act whenever a certain condition develops among its neighbors. The action is simple, since it involves only local perturbation of the processor's parameters. Both the activation and the action are meaningful because they engage semantically related propositions. Thus, whenever a problem can be solved by a constraint propagation mechanism, it is also easy to formulate in a pure production-rule formalism.

4.1.2 Why Probabilistic Reasoning Seems to Resist Propagation

While constraint propagation mechanisms have been essential to many AI applications, e.g., vision [Rosenfeld, Hummel, and Zucker 1976; Waltz 1972] and truth maintenance [McAllester 1980], their use in evidential reasoning, surprisingly, has been limited to non-Bayesian formalisms [Lowrance 1982; Quinlan 1983; Shastri and Feldman 1984]. There have been several reasons for this, all based on the essential difference between the probabilistic statement $P(A \mid B) = p$ and the logical rule $B \rightarrow A$ (see Sections 1.1.4, 1.3.1, and 2.3.1).

First, the conditional probabilities characterizing the links in the network do not seem to impose definitive constraints on the probabilities that can be assigned to the nodes. Consider a pair of nodes A and B linked by an arrow $B \rightarrow A$ and quantified by the conditional probabilities $P(a \mid b)$ and $P(a \mid \neg b)$. The quantifier $P(a \mid b)$ restricts the belief accorded to a only in a very special set of circumstances, namely, when b is known with absolute certainty to be true and when no other evidential data is available. Normally, all internal nodes in the network will be subject to some uncertainty. Thus, if processor A inspects its neighbor B and finds it in a state of uncertainty with $P(b) < 1$, it still cannot proceed with a definite action on $P(a)$. A natural recourse would be to compute the weighted average

$$P(a) = P(a \mid b)P(b) + P(a \mid \neg b)\, P(\neg b).$$

After the arrival of some evidence e, however, the posterior distributions A and B are no longer governed by $P(a \mid b)$ but rather by $P(a \mid b, e)$, via

$$P(a \mid e) = P(a \mid b, e)\, P(b \mid e) + P(a \mid \neg b, e)\, P(\neg b \mid e),$$

which may be a totally different relationship. (To take an extreme example, if e fully confirms or denies a, the overall probability of a becomes totally insensitive to $P(a \mid b)$.) The result is that any arbitrary assignment of beliefs to the propositions a and b can be consistent with the value of $P(a \mid b)$ that was initially

assigned to the link connecting them; consequently, among these parameters, no violation of a constraint can be detected locally.

Second, the disparity between $P(a \mid b, e)$ and $P(a \mid b)$ suggests that once a new piece of evidence is introduced, the original weights on the link no longer retain their intended meaning; hence, they should not remain fixed but should undergo constant adjustment as new evidence arrives. This requires enormous computational overhead and an external unit to perform the adjustment, so it defeats the whole purpose of local propagation.

Finally, the presence of both top-down *(predictive)* and bottom-up *(diagnostic)* inferences in evidential reasoning has caused apprehensions that pathological instability, deadlock, and circular reasoning will develop once we allow the propagation process to run its course unsupervised [Lowrance 1982]. Indeed, if a stronger belief in a given hypothesis means a greater expectation of the occurrence of its various manifestations, and if, in turn, a greater certainty in the occurrence of these manifestations adds further credence to the hypothesis, how can one avoid infinite updating loops when the processors responsible for these propositions begin to communicate with one another?

EXAMPLE 2: You spread a rumor about person X to your neighbor N_1. A few days later, you hear the same rumor from N_1. Should you increase your belief in the rumor now that N_1 acknowledges it, or should you determine first whether N_1 heard it from another source besides you? It is clear that if you were N_1's only source of information, your belief should not change, but if N_1 independently confirmed the validity of the rumor, you have good reason to increase your belief in it.

Similar considerations apply to communicating processors that represent interdependent propositions. Imagine that a processor F, representing the event *Fire*, communicates asynchronously with a second processor S, representing the event *Smoke*. At time t_1, some evidence (e.g., the distant sound of a fire engine) gives a slight confirmation to F, thus causing the probability of *Fire* to increase from $P(F)$ to $P_1(F)$. At a later time, t_2, processor S may decide to interrogate F; upon finding $P_1(F)$, it revises the probability of *Smoke* from $P(S)$ to $P_2(S)$ in natural anticipation of smoke. Still later, at t_3, processor F is activated, and upon finding an increased belief $P_2(S)$ in *Smoke*, it increases $P_1(F)$ to an even higher value, $P_3(F)$. This feedback process may continue indefinitely, the two processors drawing steady mutual reinforcement void of any empirical basis, until eventually the two propositions, *Fire* and *Smoke*, appear to be firmly believed.

Such dangers are not unique to probabilistic reasoning, but should be considered in any hierarchical model of cognition where mutual reinforcement takes place between lower and higher levels of processing, e.g., connectionist models of reading [Rumelhart and McClelland 1982] and language production [Dell 1985].

To prevent these phenomena, we need a mechanism to keep track of the *sources* of belief, so that evidence is not counted twice and so that the impact of one piece of evidence is not fed back to its source. Unfortunately, source identification requires an overview of the entire network, and the question arises whether it can be represented and adjusted locally as an integral part of the propagation process.

This chapter demonstrates that in a large class of networks, coherent and stable probabilistic reasoning *can* be accomplished by local propagation mechanisms, keeping the weights on the links constant throughout the process. This is done by characterizing the belief in a proposition by a list of parameters, each representing the degree of support the host proposition obtains from one of its neighbors. In the next two sections we show that maintaining such a record of the sources of belief facilitates local updating of beliefs, and that the network relaxes to a stable equilibrium, consistent with the axioms of probability theory, in time proportional to the network diameter. Such a record of parameters is also postulated as a mechanism that permits people to retrace rationales and assemble explanations for currently held beliefs.

4.2 BELIEF PROPAGATION IN CAUSAL TREES

4.2.1 Notation

We shall first consider tree-structured causal networks, i.e., those in which every node except the one called *root* has exactly one incoming link. We allow each node to represent a multi-valued variable, comprising a collection of mutually exclusive hypotheses (e.g., the identity of an organism: $Org_1, Org_2, \ldots$) or observations (e.g., a patient's temperature: *High*, *Medium*, *Low*). Let variables be labeled by capital letters ($A, B, \ldots, X, Y, Z$) and their possible values by the corresponding lowercase letters ($a, b, \ldots, x, y, z$). In dealing with propositional variables, the symbols + and $\neg$ will be used to denote the affirmation and denial, respectively, of propositions. For example, $+a$ stands for $A = TRUE$, and $\neg a$ stands for $A = FALSE$. Each directed link $X \rightarrow Y$ is quantified by a fixed conditional probability matrix M, in which the (x, y) entry is given by

$$M_{y|x} \triangleq P(y|x) \triangleq P(Y = y | X = x) \tag{4.1}$$

$$= \begin{array}{c} \\ x \downarrow \end{array} \begin{array}{c} \begin{array}{cccc} \multicolumn{4}{c}{\rightarrow y} \end{array} \\ \begin{bmatrix} P(y_1|x_1) & P(y_2|x_1) & \cdots & P(y_n|x_1) \\ P(y_1|x_2) & P(y_2|x_2) & \cdots & P(y_n|x_2) \\ \vdots & \vdots & & \vdots \\ P(y_1|x_m) & P(y_2|x_m) & \cdots & P(y_n|x_m) \end{bmatrix} \end{array}.$$

Normally, the directionality of the arrow designates X as the set of causal hypotheses and Y as the set of consequences or manifestations of these hypotheses.

EXAMPLE 3: In a certain trial there are three suspects, one of whom has definitely committed a murder. The murder weapon, showing some fingerprints, was later found by police. Let X identify the last user of the weapon, namely, the killer. Let Y identify the last holder of the weapon, i.e., the person whose fingerprints were left on the weapon, and let Z represent the possible readings that may be obtained in a fingerprint laboratory.

The relations between these three variables normally would be expressed by the chain $X \rightarrow Y \rightarrow Z$; X generates expectations about Y, and Y generates expectations about Z, but X has no influence on Z once we know the value of Y.

To represent the commonsense knowledge that the killer is normally the last person to hold the weapon, we use a 3×3 conditional probability matrix:

$$M_{y|x} = P(y|x) = \begin{cases} 0.80 \text{ if } x = y & x, y = 1, 2, 3 \\ 0.10 \text{ if } x \neq y & x, y = 1, 2, 3. \end{cases} \tag{4.2}$$

To represent the reliability of the laboratory test, we use a matrix $M_{z|y} = P(z|y)$, satisfying

$$\sum_z P(z|y) = 1 \quad \text{for } y = 1, 2, 3 .$$

Each entry in this matrix represents an if-then rule of the following type: "If the fingerprint is of Suspect y, then expect a reading of type z, with certainty $P(z|y)$."

Note that this convention is at variance with that used in many expert systems (e.g., MYCIN), where rules point from evidence to hypothesis (e.g., if symptom, then disease), thus denoting a flow of mental inference. By contrast, the arrows in Bayesian networks point from causes to effects, or from conditions to consequences, thus denoting a flow of constraints attributed to the physical world. The reason for this choice is that people often prefer to encode experiential knowledge in causal schemata [Tversky and Kahneman 1977], and as a consequence, rules expressed in causal form are assessed more reliably.†

Incoming information may be of two types: *specific evidence* or *virtual evidence*. Specific evidence corresponds to direct observations that affect the belief in some variables in the network. Virtual evidence corresponds to judgments based on undisclosed observations that are outside the network but have

† It appears that frames used to index human memory by and large are organized to evoke *expectations* rather than *explanations*. The reason could be because expectation-evoking frames normally consist of more stable relationships. For example, $P(y|z)$ in Example 3 would vary drastically with the proportion of people who have type z fingerprints. $P(z|y)$, on the other hand, depends merely on the similarity between the type of fingerprint that Suspect y has and the readings observed in the lab; it is perceived to be a stable local property of the laboratory procedure, independent of other information regarding Suspect y.

bearing on variables in the network. Such evidence is modeled by dummy nodes representing the undisclosed observations, connected by unquantified (dummy) links to the variables affected by the observations. These links will carry information one way only, from the evidence to the variables affected by it.

For example, if it is impractical for the fingerprint laboratory to disclose all possible readings (in variable Z) or if the laboratory chose to base its finding on human judgment, Z will be represented by a dummy node, and the link $Y \rightarrow Z$ will specify the degree to which each suspect is likely to bear the fingerprint pattern examined. For example, the laboratory examiner may issue a report in the form of a list (0.80, 0.60, 0.50), stating that she is 80% sure that the fingerprint belongs to Suspect 1, 60% sure that it belongs to Suspect 2, and 50% sure that it belongs to Suspect 3. If the examiner was totally unbiased before the test, such a profile of belief can be established only if the likelihood ratio is

$$P(z_{observed} \mid y = 1) : P(z_{observed} \mid y = 2) : P(z_{observed} \mid y = 3) = 8 : 6 : 5 ,$$

which will be our standard way to characterize the impact of virtual evidence (see Sections 2.2.2 and 2.3.3). Because these numbers need not sum to unity, each judgment can be formed independently of the others—each suspect's fingerprints can be compared separately with those found on the weapon.

All incoming evidence, both specific and virtual, will be denoted by $\boldsymbol{e}$ and will be regarded as emanating from a set $\boldsymbol{E}$ of *instantiated* variables, i.e., variables whose values are known. For example, if the laboratory examination is the only evidence available in Example 3, we shall write $\boldsymbol{E} = \{Z\}$ and $\boldsymbol{e} = \{Z = z_{observed}\}$. If several facts become known, say, $A = TRUE$, $B = FALSE$ and $X = x$, then $\boldsymbol{E} = \{A, B, X\}$ and $\boldsymbol{e} = \{+a, \neg b, X = x\}$.

For the sake of clarity, we will distinguish between the fixed conditional probabilities that label the links, e.g., $P(y \mid x)$, and the dynamic values of the updated node probabilities, e.g., $P(x \mid \boldsymbol{e})$. The latter will be denoted by $BEL(x)$, which reflects the overall belief accorded to proposition $X = x$ by all evidence so far received. Thus,

$$BEL(x) \triangleq P(x \mid \boldsymbol{e}) ,$$

where $\boldsymbol{e}$ is the value combination of all instantiated variables.

Since we will be dealing with discrete variables, functions such as $\lambda(x)$, $P(x)$, and $BEL(x)$ can be regarded as lists, or *vectors*, with each component corresponding to a different value of X. For example, if the domain of X is $D_X = \{Hot, Medium, Cold\}$, we can write

$$\begin{aligned} BEL(x) &= (BEL(X = Hot), BEL(X = Medium), BEL(X = Cold)) \\ &= (0.1, 0.2, 0.7) . \end{aligned}$$

The product $f(x)\, g(x)$ of two such vectors will stand for term-by-term multiplication, e.g.,

$$(1, 2, 3)\ (3, 2, 1) = (1 \times 3, 2 \times 2, 3 \times 1) = (3, 4, 3).$$

Inner products (or *dot products)* will be denoted by a dot $\bullet$, e.g.,

$$f(x) \bullet g(x) = (1, 2, 3) \bullet (3, 2, 1) = 1 \times 3 + 2 \times 2 + 3 \times 1 = 10.$$

The dot symbol $\bullet$ will also be used to indicate matrix products, e.g.,

$$f(x) \bullet M_{y|x} \triangleq \sum_x f(x)\, M_{y|x}.$$

The summation will always be taken over the repeated index, thus eliminating the need for transposing matrices or distinguishing between row and column vectors.

We shall use the symbol α to denote a *normalizing* constant, e.g.,

$$\alpha(1, 1, 3) = (0.2, 0.2, 0.6),$$

and the symbol β to denote an *arbitrary* constant, e.g.,

$$K\beta f(x) = \beta f(x) \quad \text{and}$$
$$\alpha\beta f(x) = \alpha f(x) .$$

A vector of 1s will be written $\mathbf{1}$; for example,

$$BEL(x) = \alpha \mathbf{1} = \alpha(1, 1, 1, 1) = (0.25, 0.25, 0.25, 0.25) .$$

4.2.2 Propagation in Chains

Consider the simplest of all tree-structured networks, namely, a network consisting of two nodes and a single link, $X \rightarrow Y$. If evidence $\boldsymbol{e} = \{Y = y\}$ is observed, then from Bayes' Rule, the belief distribution of X is given by

$$BEL(x) = P(x \mid \boldsymbol{e}) = \alpha\, P(x)\, \lambda(x) , \tag{4.3}$$

where $\alpha = [P(\boldsymbol{e})]^{-1}$, $P(x)$ is the prior probability of X, and $\lambda(x)$ is the likelihood vector

$$\lambda(x) = P(\boldsymbol{e} \mid x) = P(Y = y \mid x) = M_{y|x} . \tag{4.4}$$

In short, $\lambda(x)$ is simply the y's column of the link matrix M, as in Eq. (4.1). Since this matrix is stored at node Y, $\lambda(x)$ can be computed at Y and transmitted as a message to X, enabling X to compute its belief distribution $BEL(x)$.

If Y is not observed directly but is supported by indirect observation $\boldsymbol{e} = \{Z=z\}$ of a descendant Z of Y, we have the chain $X \rightarrow Y \rightarrow Z$, and we can still write

$$BEL(x) \triangleq P(x \mid \boldsymbol{e}) = \alpha P(x) \lambda(x) .$$

The likelihood vector $\lambda(x)$ can no longer be directly obtained from the matrix $M_{y|x}$, however, but must reflect the matrix $M_{z|y}$ as well. Conditioning and summing on the values of Y, we can write

$$\lambda(x) = P(\boldsymbol{e} \mid x) = \sum_y P(\boldsymbol{e} \mid y, x) P(y \mid x) = \sum_y P(\boldsymbol{e} \mid y) P(y \mid x)$$

$$= M_{y|x} \cdot \lambda(y), \tag{4.5}$$

using the fact that Y separates X from Z. Thus, we have shown that node X can still calculate its likelihood vector $\lambda(x)$ if it somehow gains access to the vector $\lambda(y)$.

Generalizing to the chain of Figure 4.2, every node can calculate the correct current value of its λ vector if it learns the correct λ vector of its successor.

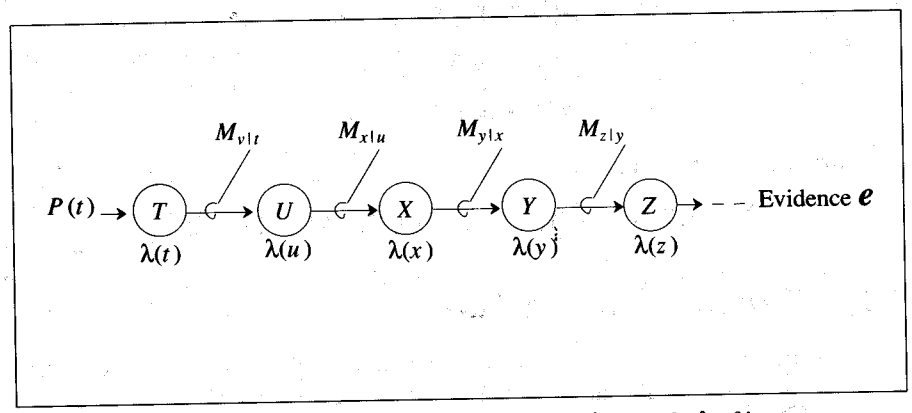

Figure 4.2. *Each node in a causal chain can calculate its λ from the λ of its successor.*

Since the chain ends with an observed variable whose value is determined externally, the λ vector of all variables can be determined recursively. If the chain ends with an unobserved variable Z and we set $\lambda(z) = 1$ for all z, Eqs. (4.3) and (4.5) are still valid, because every variable will obtain $\lambda = \mathbf{1}$ and all belief distributions will coincide with the prior distributions. Assuming that each node constantly inspects the λ of its child and updates its own λ accordingly, we are guaranteed that every variable along the chain will obtain its correct λ, properly reflecting any changes that might have occurred in $\boldsymbol{e}$. This updating process is analogous to the way soldiers are counted.

EXAMPLE 4: A platoon of soldiers is marching at night in enemy territory. The leader wants to know how many soldiers remain under his command. He sends a "count" signal to the soldier behind him. This person, in turn, looks behind, and if someone is there, he passes on the "count" signal; if no one is left behind him, he returns the signal "1" to the soldier in front of him. The soldier in front receives the "1," adds 1 (for himself) and sends "2" to the soldier in front of him, and so on. The leader eventually receives the correct count. (See Figure 4.3.) In fact, the leader need not be at the head of the platoon. He can initiate a "count" command to both his front and his back, wait for responses from both sides, and add the values received (see Figure 4.4).

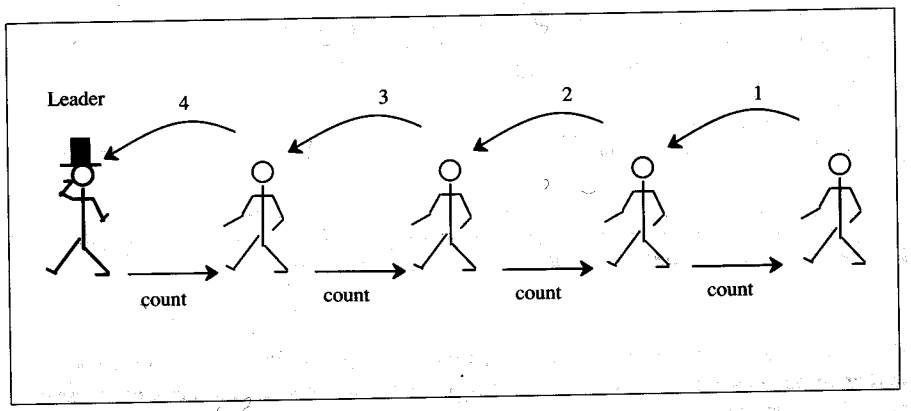

Figure 4.3. *Distributed soldier-counting.*

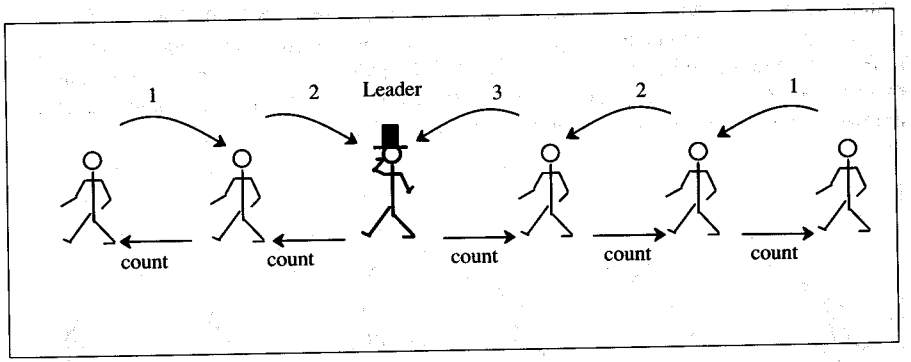

Figure 4.4. *Distributed soldier-counting with leader in line.*

This common procedure suffers because the leader may not be aware of any missing soldiers until he decides to count, and even then the count gets back to him only after communication delays to and from the end of the line. This problem can be overcome by instructing each soldier to constantly update and communicate the messages without waiting for the "count" signal, or more efficiently, to initiate communication as soon as a

change is seen in the immediate environment. Thus, no communication takes place under normal conditions, but when any soldier suddenly finds himself at either the front or the end of the line, he will initiate a message-passing process that propagates toward the leader and eventually terminates at the leader (who is now passive) with the correct count (see Figure 4.5).

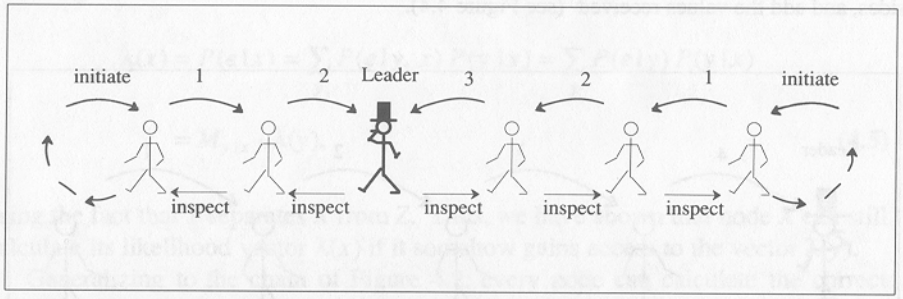

Figure 4.5. *Soldier-counting initiated by changes at endpoints.*

To draw the analogy closer to belief updating, let's remove the leader and force every soldier to be constantly aware of the current total count. In such a system, the messages, instead of stopping at any particular individual, should continue to propagate toward the periphery; the forward-moving messages should propagate all the way to the front of the line, and the backward-moving messages should propagate to the end of the line. Every soldier follows the same rule: receive a count from the person behind, add 1, and transmit the result to the person in front; receive a count from the person in front, add 1, and transmit to the one behind (see Figure 4.6). Note that each soldier must maintain and communicate two separate parameters, the back count and the front count; the overall count (the sum of the two) is not a message that can sustain the propagation.

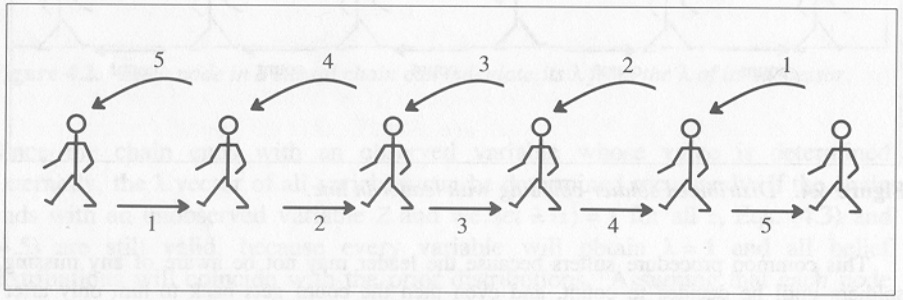

Figure 4.6. *Leaderless soldier-counting using dual-parameter communication.*

BIDIRECTIONAL PROPAGATION

The need for dual-parameter communication also exists in belief updating, where new evidence can emerge from both a descendant of a node and its ancestor. For example, in the chain of Figure 4.7, variables are instantiated at both the head and the tail of the chain, and we may wish to calculate $BEL(x)$ as a function of the values, e^+ and e^-, that these variables take.†

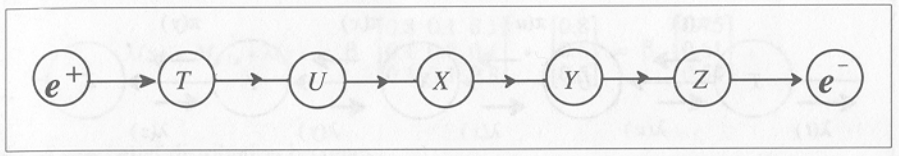

Figure 4.7. *A causal chain with evidential data at its head (e^+) and tail (e^-).*

In Eq. (4.3), $\lambda(x)$ was defined by $P(\boldsymbol{e}|x)$, $\boldsymbol{e}$ being the total evidence available. We now find it more convenient to handle the impact of e^- and e^+ by two separate vectors,

$$\lambda(x) = P(e^- \mid x) \tag{4.6a}$$

and

$$\pi(x) = P(x \mid e^+) \,. \tag{4.6b}$$

Expressing the total belief distribution $BEL(x)$ in terms of $\lambda(x)$ and $\pi(x)$, with X separating e^+ from e^-, we have

$$\begin{aligned} BEL(x) &\triangleq P(x \mid e^+, e^-) = \alpha\, P(e^- \mid x, e^+)\, P(x \mid e^+) = \alpha\, P(e^- \mid x)\, P(x \mid e^+) \\ &= \alpha\, \lambda(x)\, \pi(x) \,, \end{aligned}$$

which is identical to Eq. (4.3) with $\pi(x)$ replacing $P(x)$.

To illustrate how information about $\pi(x)$ propagates from e^+ down the chain, let us condition Eq. (4.6*b*) on the values of the parent variable U:

$$\pi(x) = P(x \mid e^+) = \sum_u P(x \mid u, e^+)\, P(u \mid e^+) \,.$$

Since U separates X from e^+, we obtain

$$\pi(x) = \sum_u P(x \mid u)\, \pi(u) = \pi(u) \bullet M_{x|u} \,. \tag{4.7}$$

† e^+ might represent the background knowledge one has about T, in which case the prior probability $P(t)$ will be identical to $P(t \mid e^+)$.

We see that the forward propagation of π's parameters is similar to the backward propagation of λ's parameters; both involve vector multiplication by the appropriate link matrix. Each node can now compute its own π and λ after obtaining the π of its parent and the λ of its child. (See Figure 4.8.)

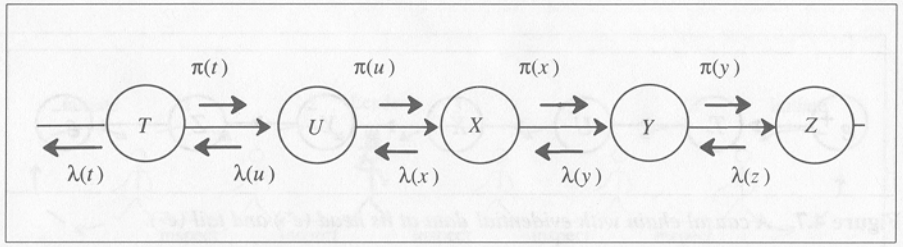

Figure 4.8. *Belief calculation using bidirected message passing in causal chains.*

EXAMPLE 5: Referring to the trial story of Example 3, let $e^- = \{Z = z\}$ represent the experience of examining the fingerprints left on the murder weapon, and let e^+ stand for all other testimony heard in the trial. So, $\pi(x) = P(x \mid e^+)$ stands for our prior certainty that the x-th suspect is the killer, $\pi(y) = P(y \mid e^+)$ stands for our prior certainty (before examining the fingerprints) that the y-th suspect was the last person to hold the weapon, and $\lambda(y) = P(e^- \mid y)$ represents the report issued by the fingerprint laboratory. Taking $\pi(x) = (0.8, 0.1, 0.1)$ and using the matrix of Eq. (4.2), we get

$$\pi(y) = (0.8, 0.1, 0.1) \bullet \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} = (0.66, 0.17, 0.17)\,.$$

Prior to inspection of the fingerprints, all λs are unit vectors 1 and the message profile on the chain is as shown in Figure 4.9.

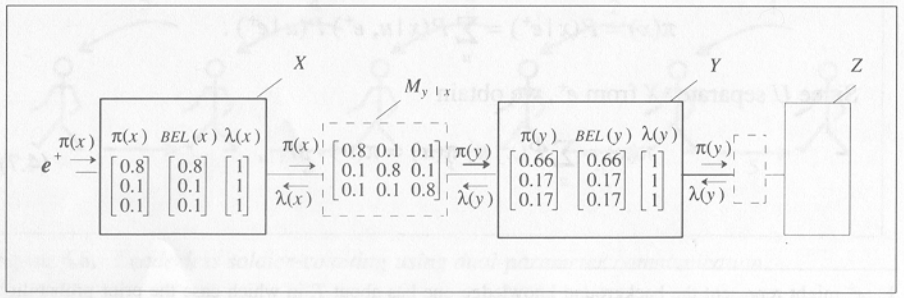

Figure 4.9. *Initial beliefs and messages in Example 5.*

Now assume that a laboratory report arrives, summarized by $\lambda(y) = \beta(0.8, 0.6, 0.5)$. Node Y updates its belief to read

$$BEL(y) = \alpha\,\lambda(y)\,\pi(y) = \alpha(0.8, 0.6, 0.5)\,(0.66, 0.17, 0.17)$$
$$= (0.738, 0.143, 0.119)$$

and delivers to X its $\lambda(y)$ vector. Upon receiving this message, node X computes its new $\lambda(x)$ vector,

$$\lambda(x) = M_{y|x} \bullet \lambda(y) = \beta \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \bullet \begin{bmatrix} 0.8 \\ 0.6 \\ 0.5 \end{bmatrix} = \beta \begin{bmatrix} 0.75 \\ 0.61 \\ 0.54 \end{bmatrix},$$

and its new belief distribution becomes

$$BEL(x) = \alpha\,\lambda(x)\,\pi(x) = \alpha(0.75, 0.61, 0.54)\,(0.8, 0.1, 0.1)$$
$$= (0.840, 0.085, 0.076)\,.$$

The messages are distributed as in Figure 4.10.

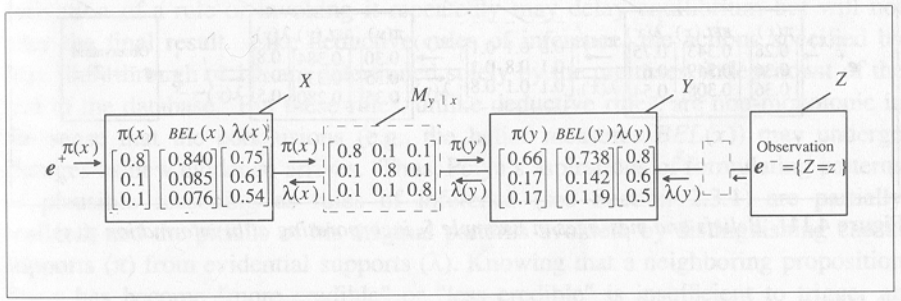

Figure 4.10. *Beliefs and messages in Example 5, after obtaining the laboratory report* $\lambda(y)$.

Now assume that Suspect 1 produces a very strong alibi, which (discounting fingerprint information) reduces the odds that he could have committed the crime from 0.80 to 0.28, thus yielding a revised prior probability of

$$\pi(x) = (0.28, 0.36, 0.36)\,.$$

This change propagates toward Y and results in a revised $\pi(y)$:

$$\pi(y) = \pi(x) \bullet M_{y|x} = (0.28, 0.36, 0.36) \bullet \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} = (0.30, 0.35, 0.35)\,.$$

Each processor can now compute the revised total belief of its variable, taking into account the evidential impact of the fingerprint findings:

$$BEL(x) = \alpha\pi(x)\lambda(x) = \alpha(0.28, 0.36, 0.36)\,(0.75, 0.61, 0.54)$$
$$= \alpha(0.210, 0.220, 0.194)$$
$$= (0.337, 0.352, 0.311)\,,$$
$$BEL(y) = \alpha\pi(y)\lambda(y) = \alpha(0.30, 0.35, 0.35)\,(0.80, 0.60, 0.50)$$
$$= \alpha(0.240, 0.210, 0.175)$$
$$= (0.384, 0.336, 0.280)\,.$$

Thus, Suspect 2 now becomes the strongest candidate for being the killer (with $P(X = 2) = 0.349$), though Suspect 1 is still most likely to be the owner of the fingerprint (with $P(Y = 1) = 0.384$). The final message distribution is shown in Figure 4.11.

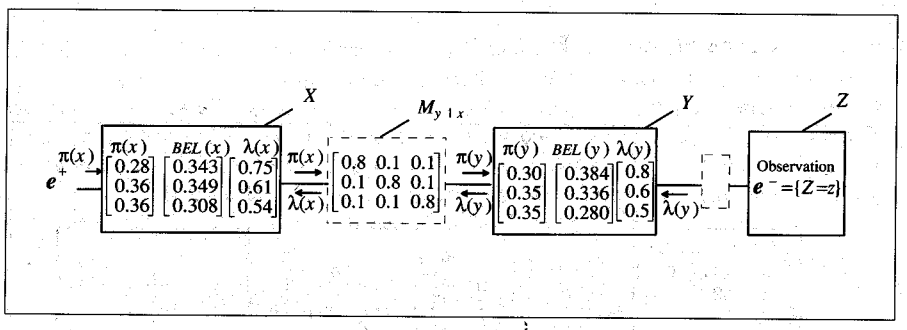

Figure 4.11. *Beliefs and messages in Example 5, incorporating alibi information* $\pi(x)$.

Note how the separation between causal and evidential support (i.e., between the πs and the λs) prevents feedback, circular reasoning, and indefinite relaxations of the type discussed in Section 4.1.1. Suspect 1's alibi renders him less likely (by a factor of 0.384/0.738) to be the owner of the incriminating fingerprints, but this reduction is not fed back to further influence his likelihood of being the killer (this would amount to counting the alibi information twice) because $\lambda(x)$ and $\lambda(y)$ are unaffected by changes occurring in $\pi(x)$. Keeping these two modes of support orthogonal to each other ensures stable updating in a single pass.

The local computations performed by each processor are essentially constraint relaxations of the type discussed in Example 1, with two additional features: the constraints are equalities, and they can always be satisfied locally. If the state of each processor is defined by its associated π and λ vectors, then the updating procedure can be written as a collection of local inference rules, identical in form and spirit to those used in constraint relaxation and logical deduction. For

example, assuming the content of π and λ is stored in two registers, called Π and Λ, the behavior of processor X in Figure 4.8 is specified by three inference rules:

$$\text{If } (X \to Y)_{M_{y|x}} \text{ and } \Lambda(Y) = \lambda(y) \quad \text{then } \Lambda(X) = \lambda(y) \cdot M_{y|x}\,, \tag{4.8}$$

$$\text{If } (U \to X)_{M_{x|u}} \text{ and } \Pi(U) = \pi(u) \quad \text{then } \Pi(X) = M_{x|u} \cdot \pi(u)\,, \tag{4.9}$$

$$\text{If } \Lambda(X) = \lambda(x) \text{ and } \Pi(X) = \pi(x) \quad \text{then } BEL(x) = \alpha\lambda(x)\,\pi(x)\,. \tag{4.10}$$

The first rule, for instance, reads:

> **If** the rule "If $X = x$ then $Y = y$" was assigned the certainty $M_{y|x}$, **and** the current content of $\Lambda(Y)$ is $\lambda(y)$, **then** put $\sum_y \lambda(y)\, M_{y|x}$ into $\Lambda(X)$.

Eqs. (4.8) through (4.10) are depicted in Figure 4.12. The reasons for formulating simple updating equations like Eqs. (4.5) and (4.7) as *inference rules* is to demonstrate their similarity to logical deductions. Like deductive rules of inference, they can be invoked at any time and in any order; postponing the activation of a rule or invoking it repeatedly may delay equilibrium but will not alter the final result. Like deductive rules of inference, the actions specified by Eqs. (4.8) through (4.10) are determined solely by the premises, independent of the rest of the database. But these rules, unlike deductive rules, are non-monotonic in the sense that the conclusions (e.g., the belief measures $BEL(x)$) may undergo changes as new evidence arrives. Thus, Polya's aspirations of formulating patterns of plausible reasoning as rules of inference (see Section 2.3.1) are partially realized, and the pitfalls of his original patterns avoided, by distinguishing causal supports (π) from evidential supports (λ). Knowing that a neighboring proposition $Y = y$ has become "more credible" or "less credible" is insufficient to trigger an action in X; we must first ascertain whether it is $\lambda(y)$ or $\pi(y)$ that has changed. This distinction is expressed in logical terms in Section 10.3.

The scheme described in Figures 4.9 through 4.12 requires that each processor gain access to matrices of both incoming and outgoing links. This may be inconvenient both for hardware implementation and rule-based programming. An alternative scheme, depicted in Figure 4.13, requires that each processor store just one matrix corresponding to the incoming link. Here, each processor receives as input the π of its parent and the λ of its own variable. Upon activation, each processor computes its own π (to be delivered to its child) and the λ of its parent (to be delivered to the parent). This convention will be used throughout the rest of this chapter because it closely reflects the basic construction of Bayesian networks, whereby each node is characterized by its relation to its parents.

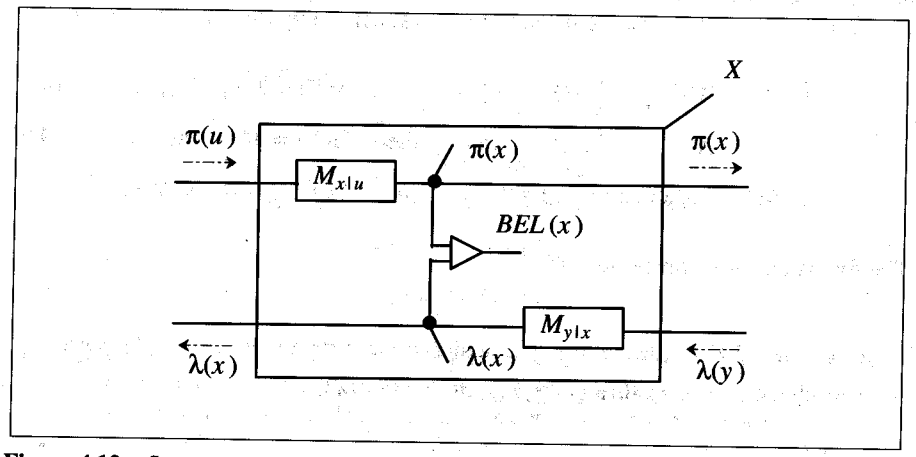

Figure 4.12. *Structure of individual processor, containing two link matrices.*

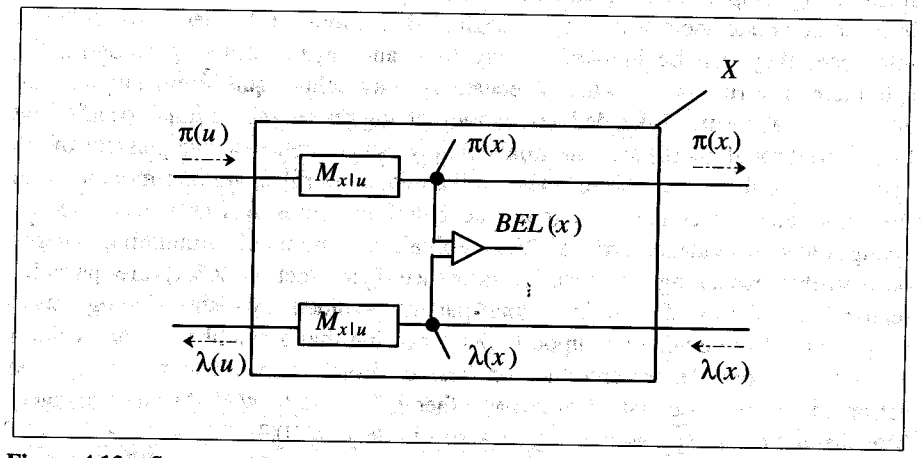

Figure 4.13. *Structure of individual processor, containing a single link matrix.*

4.2.3 Propagation in Trees

We now examine a general tree-structured network where a node might have several children and one parent. The propagation scheme in trees is very similar to that of chains, with two distinctions: Each node must combine the impacts of λ-messages obtained from several children, and each node should distribute a separate π-message to each of its children.

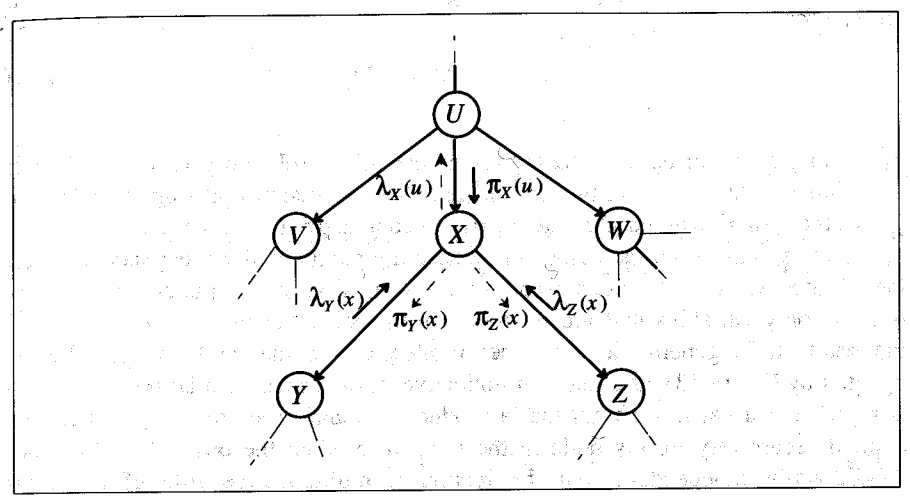

Figure 4.14. *Fragment of causal tree, showing incoming (solid arrows) and outgoing (broken arrows) messages at node X.*

Consider the tree fragment depicted in Figure 4.14. The belief in the various values of X depends on two distinct sets of evidence: evidence from the sub-tree rooted at X, and evidence from the rest of the tree. But the influence of the latter source of information on X is completely summarized by its effect on U, since U separates X from all variables except X's descendants. More formally, letting e_X^- stand for the evidence contained in the tree rooted at X and letting e_X^+ stand for the evidence contained in the rest of the network, we have

$$P(x \mid u, e_X^+) = P(x \mid u)\,. \tag{4.11}$$

This also leads to the usual conditional independence among siblings,

$$P(x, v \mid u) = P(x \mid u)\,P(v \mid u)\,, \tag{4.12}$$

since the proposition $V = v$ is part of e_X^+.

DATA FUSION

Assume we wish to find the belief induced on X by some evidence $e = e_X^- \cup e_X^+$. Bayes' Rule, together with Eq. (4.11), yields the product rule

$$\begin{aligned} BEL(x) &= P(x \mid e_X^+, e_X^-) = \alpha\, P(e_X^- \mid e_X^+, x)\, P(x \mid e_X^+) \\ &= \alpha\, P(e_X^- \mid x)\, P(x \mid e_X^+)\,, \end{aligned} \tag{4.13}$$

where $\alpha = [P(e_X^- \mid e_X^+)]^{-1}$ is a normalizing constant.

Eq. (4.13) provides an interesting generalization of the celebrated Bayes product formula,

$$P(x \mid e) = \alpha P(e \mid x) P(x), \tag{4.14}$$

by identifying a surrogate—$P(x \mid e_X^+)$—for the prior probability term $P(x)$—with every intermediate node in the tree. In recursive Bayesian updating (see Section 2.1.4), the posterior probability can be used as a new prior, relative to the next item of evidence, only when the items of evidence are conditionally independent, given the updated variable X. Such recursive updating cannot be applied to networks because only variables that are separated from each other by X are conditionally independent. In general, it is not permissible to use the total posterior belief, updated by Eq. (4.13), as a new multiplicative prior for the calculation. Eq. (4.13) is significant because it shows that a product rule analogous to Eq. (4.14) can be applied recursively to any node in the tree, even when the observations are not conditionally independent, but the recursive, multiplicative role of the prior probability has been taken over by that portion of belief contributed by evidence from the sub-tree above the updated variable, excluding the data collected from its descendants. The root is the only node that requires a prior probability estimation, and since it has no network above it, e_{root}^+ should be interpreted as the background knowledge remaining unexplicated.

Eq. (4.13) suggests that the probability distribution of every variable in the tree can be computed if the node corresponding to that variable contains the vectors

$$\lambda(x) = P(e_X^- \mid x) \tag{4.15}$$

and

$$\pi(x) = P(x \mid e_X^+) . \tag{4.16}$$

Here, $\pi(x)$ represents the causal or *predictive* support attributed to the assertion "$X = x$" by all non-descendants of X, mediated by X's parent, and $\lambda(x)$ represents the diagnostic or *retrospective* support that "$X = x$" receives from X's descendants. The total strength of belief in "$X = x$" can be obtained by *fusing* these two supports via the product

$$BEL(x) = \alpha\lambda(x)\,\pi(x). \tag{4.17}$$

To see how information from several descendants fuses at node X, we partition the data set e_X^- in Eq. (4.15) into disjoint subsets, one for each child of X. Referring to Figure 4.14, for example, the tree rooted at X can be partitioned into the root, X,

and two sub-trees, one rooted at Y and the other at Z. Thus, if X itself is not instantiated, we can write $e_X^- = e_Y^- \cup e_Z^-$, and since X separates its children, we have

$$\begin{aligned} \lambda(x) &= P(e_X^- \mid x) \\ &= P(e_Y^-, e_Z^- \mid x) \\ &= P(e_Y^- \mid x) P(e_Z^- \mid x) . \end{aligned} \tag{4.18}$$

So $\lambda(x)$ can be formed as a product of terms such as $P(e_Y^- \mid x)$, if these terms are delivered to X as messages from its children. Denoting these messages by subscripted λ's,

$$\lambda_Y(x) = P(e_Y^- \mid x) \tag{4.19a}$$

and

$$\lambda_Z(x) = P(e_Z^- \mid x) , \tag{4.19b}$$

we have the product rule:

$$\lambda(x) = \lambda_Y(x)\,\lambda_Z(x) . \tag{4.20}$$

This product rule also applies when X itself is instantiated ($X = x'$) if we model the new data by adding to X a dummy child D that delivers the message

$$\lambda_D(x) = \delta_{x, x'} = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{if } x \neq x'. \end{cases}$$

EXAMPLE 6: In the fingerprint story of Example 5, imagine that we receive reports from two independent laboratories, $\lambda_{Z_1}(y) = \beta$ (0.80, 0.60, 0.50) and $\lambda_{Z_2}(y) = \beta$ (0.30, 0.50, 0.90). The overall diagnostic support $\lambda(y)$ attributable to the three possible values of Y is

$$\lambda(y) = \beta(0.80, 0.60, 0.50)\,(0.30, 0.50, 0.90) = \beta(0.24, 0.30, 0.45) ,$$

and this, combined with the previous causal support $\pi(y) = (0.30, 0.35, 0.35)$, yields an overall belief of

$$\begin{aligned} BEL(y) &= \alpha(0.24, 0.30, 0.45)\,(0.30, 0.35, 0.35) \\ &= (0.215, 0.314, 0.471) . \end{aligned}$$

What happens if Suspect 2 confesses, reliably, that he was the last weapon holder? We model this confession as a third report $\lambda_{Z_3}(y) = (0, 1, 0)$ which, by the product rule of Eq. (4.20), completely overrides the other two and yields $\lambda(y) = \beta(0, 1, 0)$ and $BEL(y) = (0, 1, 0)$.

Now we shall see if X can compute its $\pi(x)$ vector from information available at its parent U (see Figure 4.14).

Conditioning on the values of U we get

$$\begin{aligned}\pi(x) &= P(x \mid e_X^+) \\ &= \sum_u P(x \mid e_X^+, u) P(u \mid e_X^+) \\ &= \sum_u P(x \mid u) P(u \mid e_X^+) .\end{aligned}$$

$P(x \mid u)$ is the matrix stored on the link $U \rightarrow X$, and $P(u \mid e_X^+)$ can be calculated by U and delivered to X as the message

$$\pi_X(u) = P(u \mid e_X^+) , \tag{4.21}$$

yielding

$$\pi(x) = \sum_u P(x \mid u) \pi_X(u) = M_{x \mid u} \bullet \pi_X(u). \tag{4.22}$$

Substituting Eqs. (4.20) and (4.22) in Eq. (4.17) we have

$$BEL(x) = \alpha \lambda_Y(x) \lambda_Z(x) \sum_u P(x \mid u) \pi_X(u) . \tag{4.23}$$

Thus, node X can calculate its own beliefs if it has received the messages $\lambda_Y(x)$ and $\lambda_Z(x)$ from its children Y and Z and the message $\pi_X(u)$ from its parent U.

PROPAGATION MECHANISM

Our next task is to determine how the influence of new information will spread through the network. In other words, we imagine that each node eventually receives from its neighbors the π–λ messages needed to calculate its own belief, as in Eq. (4.23), and we must determine how that node calculates the π–λ messages that its neighbors expect to receive from it. If the calculations can be accomplished by local computations, and if we let each node perform the calculations often enough, then the proper belief distributions are guaranteed to be reached, eventually, at every node.

Consider first the message $\lambda_X(u) \triangleq P(e_X^- \mid u)$ that node X must send to its parent U (see Figure 4.14). If the value of X is known, say $X = x'$, then $\lambda_X(u)$ reduces to

$P(x' \mid u)$, which is column x' of the matrix $M_{x \mid u}$. If X is not known, we condition $P(e_X^- \mid u)$ on $X = x$ and get

$$\begin{aligned}\lambda_X(u) &= \sum_x P(e_X^- \mid u, x) P(x \mid u) \\ &= \sum_x P(e_X^- \mid x) P(x \mid u) \\ &= \sum_x \lambda(x) P(x \mid u) \\ &= M_{x \mid u} \bullet \lambda(x).\end{aligned} \tag{4.24}$$

Thus, the message going to the parent U can be calculated from the messages received from the children and the matrix stored on the link from the parent. Note that Eq. (4.24) also holds if X itself is instantiated (say to $X = x'$) because in such a case $\lambda(x) = \delta_{x, x'}$, and Eq. (4.24) yields $\lambda_X(u) = P(x' \mid u)$ as required.

Now, consider the message that node X should send to one of its children, say Y:

$$\pi_Y(x) = P(x \mid e_Y^+) = P(x \mid e_X^+, e_Z^-) .$$

Using Bayes' Rule, we get

$$\begin{aligned}\pi_Y(x) &= \alpha P(e_Z^- \mid x, e_X^+) P(x \mid e_X^+) \\ &= \alpha P(e_Z^- \mid x) P(x \mid e_X^+) \\ &= \alpha \lambda_Z(x) \pi(x) \\ &= \alpha \lambda_Z(x) \sum_u P(x \mid u) \pi_X(u) .\end{aligned} \tag{4.25}$$

The second equality follows from the fact that X separates e_Z^- from e_X^+, the third equality follows from the definition of $\pi(x)$ (Eq. (4.16)), and the fourth follows from Eq. (4.22). Thus, the message sent from X to Y is calculated using the message it receives from its *other child* Z (in general, the messages it receives from all its children, except Y) and the message X receives from its parent U. This is precisely how double-counting of evidence is prevented.

Figure 4.15 summarizes the calculations for node X.

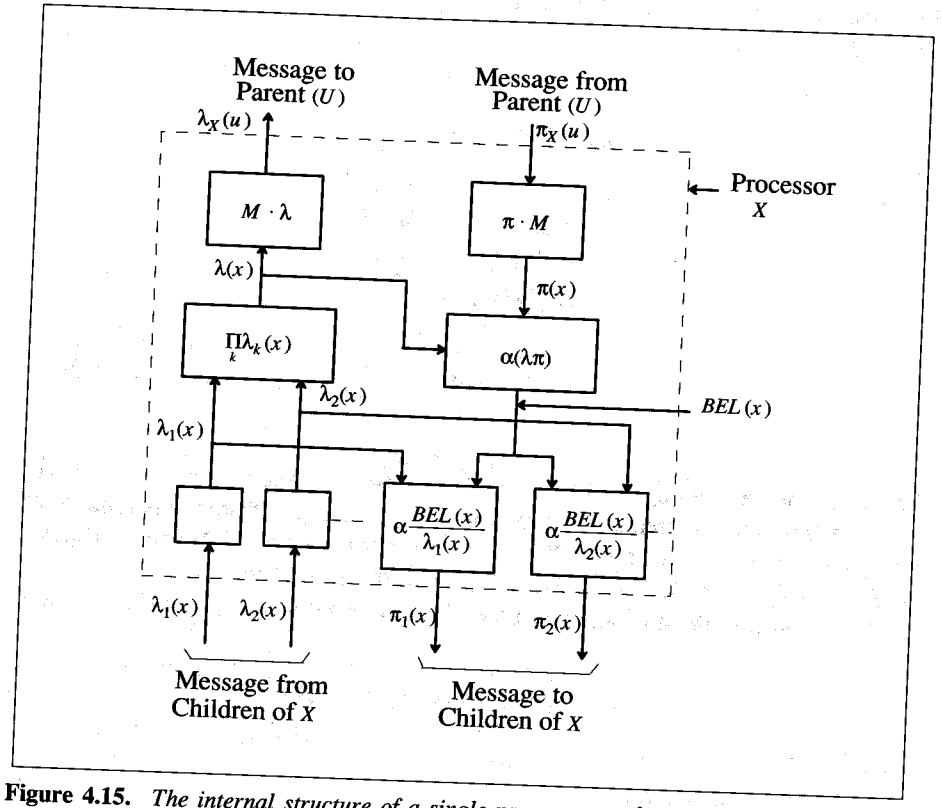

Figure 4.15. *The internal structure of a single processor performing belief updating for the variable X.*

By Eqs. (4.23) and (4.25),

$$\pi_Y(x) = \alpha \frac{BEL(x)}{\lambda_Y(x)}, \tag{4.26}$$

so it might be advantageous for node X, instead of sending each child a different message, to send all its children the value of its current belief, $BEL(x)$, and let each child recover its respective π message by dividing $BEL(x)$ by the value of the last message sent to X (caution should be exercised whenever $\lambda(x)$ is zero or is very close to zero). There is no need, of course, to normalize the π messages prior to transmission; only the $BEL(\cdot)$ expressions require normalization. The sole purpose of the normalization constant α in Eqs. (4.25) and (4.26) is to preserve the probabilistic meaning of these messages. It is a good engineering practice to encode the π and λ messages so that the smallest component of each will attain the value 1.

SUMMARY OF PROPAGATION RULES FOR TREES

We shall now summarize the steps involved in tree propagation by specifying the activities of a typical node X having m children, $Y_1, Y_2, \ldots, Y_m$, and a parent U. The belief distribution of variable X can be computed if three types of parameters are made available:

1. The current strength of the *causal* support, $\pi_X(u)$, contributed by the parent of X,
$$\pi_X(u) = P(u \mid e_X^+).$$
2. The current strength of the *diagnostic* support, $\lambda_{Y_j}(x)$, contributed by the j-th child of X,
$$\lambda_{Y_j}(x) = P(e_{Y_j}^- \mid x).$$
3. The fixed conditional probability matrix $P(x \mid u)$ that relates the variable X to its immediate parent U.

Using these parameters, local belief updating can be accomplished in three steps, to be executed in any order.

Step 1—Belief updating: When node X is activated to update its parameters, it simultaneously inspects the $\pi_X(u)$ message communicated by its parent and the messages $\lambda_{Y_1}(x), \lambda_{Y_2}(x), \ldots$ communicated by each of its children. Using this input, it updates its belief measure to

$$BEL(x) = \alpha\, \lambda(x)\, \pi(x), \tag{4.27a}$$

where

$$\lambda(x) = \prod_j \lambda_{Y_j}(x), \tag{4.27b}$$

$$\pi(x) = \sum_u P(x \mid u)\, \pi_X(u), \tag{4.27c}$$

and α is a normalizing constant rendering $\sum_x BEL(x) = 1$.

Step 2—Bottom-up propagation: Using the λ messages received, node X computes a new message, $\lambda_X(u)$, which is sent to its parent U:

$$\lambda_X(u) = \sum_x \lambda(x)\, P(x \mid u). \tag{4.28}$$

Step 3—Top-down propagation: X computes new π messages to be sent to each of its children. For example, the new $\pi_{Y_j}(x)$ message that X sends to its j-th child Y_j is computed by

$$\pi_{Y_j}(x) = \alpha\, \pi(x) \prod_{k \neq j} \lambda_{Y_k}(x)\,. \tag{4.29}$$

The computations in Eqs. (4.27), (4.28), and (4.29) preserve the probabilistic meaning of the parameters. In particular,

$$\lambda_X(u) = P(e_X^- \mid u)\,, \tag{4.30}$$

$$\pi_Y(x) = P(x \mid e_Y^+)\,, \tag{4.31}$$

$$BEL(x) = P(x \mid e)\,. \tag{4.32}$$

Terminal and evidence nodes in the tree require special treatment. We must distinguish four cases:

1. *Anticipatory node*—a leaf node that has not been instantiated. For such variables, *BEL* should be equal to π, and we should therefore set λ=(1,1,...,1).

2. *Evidence node*—a variable with instantiated value. Following Eq. (4.6*a*), if the j-th value of X is observed to be true, we set $\lambda(x)$ =(0,..., 0,1,0,..., 0) with 1 at the j-th position.

3. *Dummy node*—a node Y representing virtual or judgmental evidence bearing on X. We do not specify $\lambda(y)$ or $\pi(y)$ but instead post a $\lambda_Y(x)$ message to X, where $\lambda_Y(x) = \beta\, P(Observation \mid x)$, β being any convenient constant.

4. *Root node*—The boundary condition for the root node is established by setting π (root) equal to the prior probability of the root variable.

EXAMPLE 7: To illustrate these computations let us redo Example 5, using tree propagation on the network of Figure 4.16.

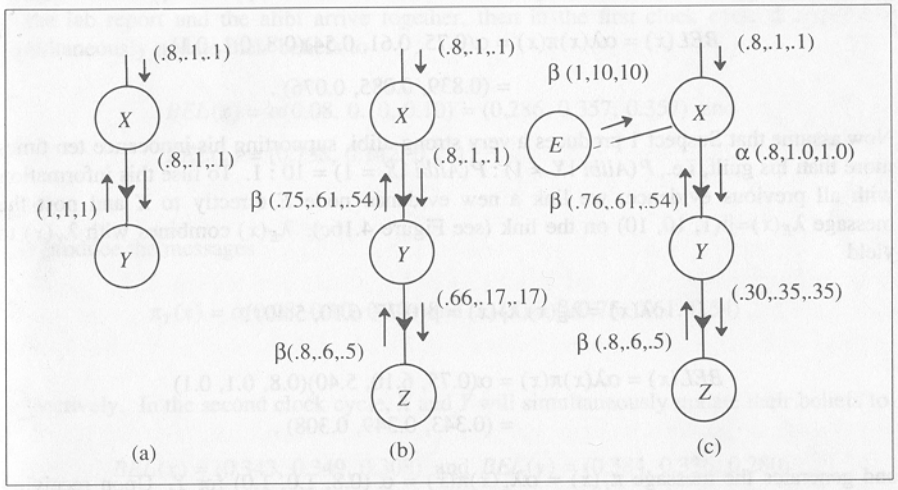

Figure 4.16. *Belief updating in Example 7 using tree propagation. The alibi is modeled as a dummy node E generating a* λ *message 1:10:10.*

As before, let us assume that our belief in the identity of the killer, based on all testimony heard so far, amounts to $\pi(x) = (0.8, 0.1, 0.1)$. Before we obtain any fingerprint information, Figure 4.16*a* shows Y as an anticipatory node with $\lambda(y) = (1, 1, 1)$, which means $\lambda_Y(x) = \lambda(x) = (1, 1, 1)$ and $BEL(x) = \pi(x)$. $\pi(y)$ can be calculated from Eq. (4.22) (using $\pi_Y(x) = \pi(x)$), yielding

$$\pi(y) = \pi_Y(x) \bullet M_{y|x} = (0.8, 0.1, 0.1) \bullet \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

$$= (0.66, 0.17, 0.17) = BEL(y)\,.$$

Assume that a laboratory report arrives summarizing the test results (a piece of virtual evidence Z) by the message $\lambda_Z(y) = \lambda(y) = \beta(0.8, 0.6, 0.5)$, as in Figure 4.16*b*. Node Y updates its belief,

$$BEL(y) = \alpha\lambda(y)\pi(y) = \alpha(0.8, 0.6, 0.5)(0.66, 0.17, 0.17) = (0.738, 0.143, 0.119)\,,$$

and based on Eq. (4.28), Y computes a new message, $\lambda_Y(x)$, for X:

$$\lambda_Y(x) = M_{y|x} \bullet \lambda(y) = \beta \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \bullet \begin{bmatrix} 0.8 \\ 0.6 \\ 0.5 \end{bmatrix} = \beta(0.75, 0.61, 0.54)\,.$$

Upon receiving this message, node X sets $\lambda(x) = \lambda_Y(x)$ and recomputes its belief to

$$BEL(x) = \alpha\lambda(x)\pi(x) = \alpha(0.75, 0.61, 0.54)(0.8, 0.1, 0.1)$$
$$= (0.839, 0.085, 0.076) .$$

Now assume that Suspect 1 produces a very strong alibi, supporting his innocence ten times more than his guilt, i.e., $P(Alibi \mid X \neq 1) : P(Alibi \mid X = 1) = 10 : 1$. To fuse this information with all previous evidence, we link a new evidence node E directly to X and post the message $\lambda_E(x)=\beta(1, 10, 10)$ on the link (see Figure 4.16*c*). $\lambda_E(x)$ combines with $\lambda_Y(x)$ to yield

$$\lambda(x) = \lambda_E(x)\,\lambda_Y(x) = \beta(0.75, 6.10, 5.40) ,$$

$$BEL(x) = \alpha\lambda(x)\pi(x) = \alpha(0.75, 6.10, 5.40)(0.8, 0.1, 0.1)$$
$$= (0.343, 0.349, 0.308) ,$$

and generates the message $\pi_Y(x) = \alpha\lambda_E(x)\pi(x) = \alpha\ (0.8, 1.0, 1.0)$ for Y. Upon receiving $\pi_Y(x)$, processor Y updates its causal support $\pi(y)$ to (see Eq. 4.27)

$$\pi(y) = \pi_Y(x) \cdot M_{y\mid x} = \alpha(0.8, 1.0, 1.0) \cdot \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} = (0.30, 0.35, 0.35) ,$$

and $BEL(y)$ becomes

$$BEL(y) = \alpha\lambda(y)\pi(y) = \alpha(0.8, 0.6, 0.5)(0.30, 0.35, 0.35)$$
$$= (0.384, 0.336, 0.280) .$$

Finally, since Y has only one child—Z—Eq. (4.29) reduces to $\pi_Z(y) = \pi(y)$ (see also Eq. (4.26)). The purpose of propagating beliefs top-down to sensory nodes such as Z is twofold: to guide data-acquisition strategies toward the most informative sources (see Section 6.4) and to facilitate explanations for the system's choices.

Note that $BEL(x)$ cannot be taken as an updated prior of x for the purpose of calculating $BEL(y)$. In other words, it is wrong to update $BEL(y)$ via the textbook formula

$$BEL(y) = \sum_x P(y \mid x)\, BEL(x) \qquad \textbf{(4.33)}$$

(see discussion of Jeffrey's Rule, Section 2.3.3), because $BEL(x)$ was affected by information transmitted from Y, and feeding this information back to Y would amount to counting the same evidence twice. Only the $\pi_Y(x)$ portion of $BEL(x)$ is fed back to Y; it is based on evidence (E) not yet considered in $\lambda(y)$. Another way to view this is that once information is obtained from Z, the initial value of the link matrix $P(y \mid x)$ no longer represents the dependence between X and Y, so $P(y \mid x, z)$ should replace $P(y \mid x)$ in Eq. (4.33).

Note also that the activation steps need not be sequential but may be executed in parallel when several pieces of evidence arrive simultaneously. In the extreme case, we can

imagine that all processors are activated simultaneously by a common clock. For example, if the lab report and the alibi arrive together, then in the first clock cycle X and Y will simultaneously update their beliefs to

$$BEL(x) = \alpha(0.08, 0.10, 0.10) = (0.286, 0.357, 0.357) \text{ and}$$
$$BEL(y) = (0.738, 0.142, 0.111)$$

and produce the messages

$$\pi_Y(x) = \alpha(0.08, 0.10, 0.10) \text{ and } \lambda_Y(x) = \beta(0.75, 0.61, 0.54) ,$$

respectively. In the second clock cycle, X and Y will simultaneously update their beliefs to

$$BEL(x) = (0.343, 0.349, 0.308) \text{ and } BEL(y) = (0.384, 0.336, 0.280)$$

and produce the same $\pi_Y(x)$ and $\lambda_Y(x)$ as before. From now on, the same beliefs and the same messages will be produced in every clock cycle unless additional evidence becomes available.

ILLUSTRATING THE FLOW OF BELIEF

Figure 4.17 shows six successive stages of belief propagation through a simple binary tree, assuming the updating is triggered by changes in the belief parameters of neighboring processors. Initially, the tree is in equilibrium, and all terminal nodes are anticipatory (Figure 4.17*a*). As soon as two data nodes are activated, white tokens are placed on the links from the nodes to their parents (Figure 4.17*b*). In the next phase, the parents, activated by these tokens, absorb them and manufacture enough tokens for their neighbors: white tokens for their parents and black ones for their children (Figure 4.17*c*). (The links from which the absorbed tokens originated do not receive new tokens because a π-message is not affected by a λ-message crossing the same link.) The root node now receives two white tokens, one from each of its descendants, triggering the production of two black tokens for top-down delivery (Figure 4.17*d*). The process continues in this fashion for six cycles, at which point all tokens are absorbed and the network reaches a new equilibrium. As soon as a leaf node posts a token for its parent, the leaf is ready to receive new data. If the new data arrives before the token was observed by the parent, a new token replaces the old one. In this fashion the inference network can track a changing environment and provide coherent interpretation of signals emanating simultaneously from multiple sources.

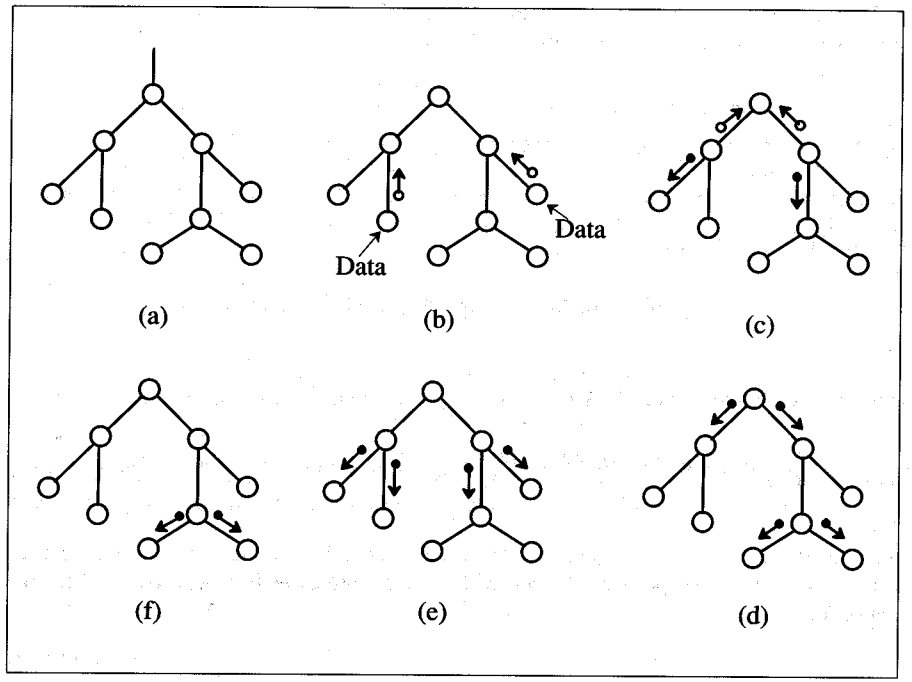

Figure 4.17. *The impact of new data propagates through a tree by a message-passing process.*

This updating scheme has the following properties:

1. The local computations it requires are efficient in both storage space and time. For a tree with m children per parent and n values per node, each processor should store $n^2+mn+2n$ real numbers and perform $2n^2+mn+2n$ multiplications per update.

2. The local computations and the final belief distribution are entirely independent of the control mechanism that activates the individual operations. These operations can be activated by either data-driven or goal-driven (e.g., requests for evidence) control strategies, by a central clock, or at random.

3. New information is diffused through the network in a single pass. Instabilities and indefinite relaxations are eliminated by maintaining a two-parameter system (π and λ) to decouple causal support from diagnostic support. The time required for completing the diffusion (in parallel) is proportional to the diameter of the network.

4.3 BELIEF PROPAGATION IN CAUSAL POLYTREES (SINGLY CONNECTED NETWORKS)

The tree structures treated in the preceding section require that exactly one variable be considered a cause of another given variable. This restriction simplifies computations, but its representational power is rather limited, since it forces us to form a single node from all causes sharing a common consequence. By contrast, when people see many potential causes for a given observation, they weigh one cause against another as independent variables, each pointing to a specialized area of knowledge. As an illustration, consider the situation discussed in Example 7 of Chapter 2:

> Mr. Holmes receives a phone call at work from his neighbor notifying him that she heard a burglar alarm sound from the direction of his home. As he is preparing to rush home, Mr. Holmes recalls that the alarm recently was triggered by an earthquake. Driving home, he hears a radio newscast reporting an earthquake 200 miles away.

Mr. Holmes perceives two episodes as potential causes for the alarm sound—an attempted burglary and an earthquake. Even though burglaries can safely be assumed independent of earthquakes, the radio announcement still reduces the likelihood of a burglary, as it "explains away" the alarm sound. Moreover, the two causal events are perceived as individual variables pointing to separate frames of knowledge (crime-related information seldom evokes associations of earthquakes), so it would be unnatural to lump the two events together into a single node.

Treating E = *Earthquake* and B = *Burglary* as two separate entities (as in Figure 2.2) allows us to relate each of them to a separate set of evidence without consulting the other. For example, if R = *The radio announcement* and S = *The alarm sound*, we can quantify the relation between E and R by the probabilities $P(R \mid E)$ without having to consider the irrelevant event of burglary, as would be required if the pair (E, B) were combined into one variable. Moreover, if R is confirmed, a natural way to update the beliefs of E and B would be in two separate steps, mediated by updating S. E and B are presumed to be independent unless evidence supporting S is obtained (e.g., the neighbor's phone call); when this happens, E and B find themselves competing for a fixed amount of evidential support—information favoring one explanation (e.g., the radio report) would weaken the other explanation by undermining its connection with the mediator S.

This competitive interplay among multiple explanations is a prevailing feature of human reasoning and has been discussed in previous chapters (see Sections 1.2.2 and 2.2.4). When a physician discovers evidence in favor of one disease, it reduces the likelihood of other diseases that could explain the patient's symptoms,

variable X, as in Figure 4.22*a*. When the global inhibitor I_0 is ON, X will be OFF, regardless of other causal factors.

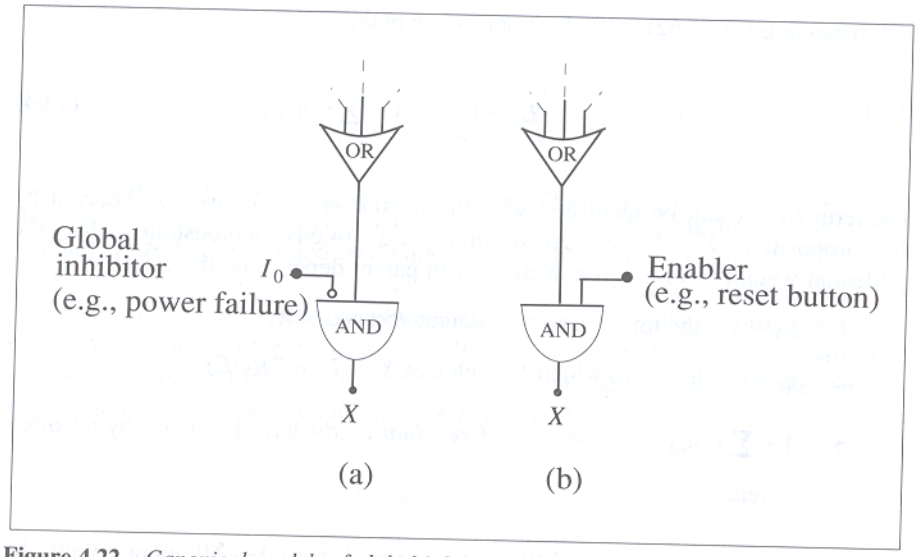

Figure 4.22. *Canonical models of global inhibition (a) and enabling mechanisms (b).*

In this fashion, we can also model various enabling mechanisms, i.e., conditions that have no influence of their own (on X) except to enable other influences to take effect. For example, if the alarm system has a reset button which Mr. Holmes occasionally forgets to push, setting this button is an enabling condition, as in Figure 4.22*b*.

SUMMARY

Canonical models can be thought of as default strategies for completing the specification of a Bayesian network whenever detailed interactions among causes are unavailable, too numerous to elicit, or too complex to be treated precisely. In particular, the disjunctive model of interacting causes has several advantages: it requires the specification of only n parameters (for a family of n parents), it executes the propagation routine in only n steps (for each node with n parents), and it leads to conclusions that match our intuition about how credit should be assigned among competing explanations. Having explicated the assumptions behind disjunctive interaction, i.e., accountability and exception independence, we can scrutinize the model's range of adequacy at a very basic level, and once this test is passed, the adequacy of the credit assignment policy is guaranteed.

4.4 COPING WITH LOOPS

Loops are undirected cycles in the *underlying* network, i.e., the network without the arrows.† When loops are present, the network is no longer singly connected, and local propagation schemes will invariably run into trouble. The reason is both architectural and semantic. If we ignore the existence of loops and permit the nodes to continue communicating with each other as if the network were singly connected, messages may circulate indefinitely around these loops, and the process may not converge to a stable equilibrium. This problem is usually encountered in non-probabilistic systems (such as Example 1), where each message can cause a discrete change in the state of the receiving processor (e.g., choosing a different color), a change that can be reversed each time its impact circulates around the loop. Such oscillations do not normally occur in probabilistic networks because of the stochastic nature of the link matrices, which tend to bring all messages toward some stable equilibrium as time goes on. However, this asymptotic equilibrium is not coherent, in the sense that it does not represent the posterior probabilities of all nodes of the network. The reason for this is simple: all of our propagation equations were based on some conditional independence assumptions that might be violated in multiply connected networks. For example, the product in Eq. (4.51) was based on the assumption that all parents of a node X are mutually independent as long as none of their common descendants is instantiated. This assumption will no longer be valid if some parents of X share a common ancestor. Even our basic fusion equation (Eq. (4.13)) was based on a clear distinction between causal and diagnostic evidence, the two being separated by X; the distinction gets blurred in multiply connected networks because $e_{\bar{X}}$ may influence X's parent via pathways that sidestep X. (Asymptotic relaxation can still be used as a method of approximation. See Exercise 4.7.)

This section introduces three coherent methods of handling loops while retaining some of the flavor of local computation: clustering, conditioning, and stochastic simulation. *Clustering* involves forming compound variables in such a way that the resulting network of clusters is singly connected. *Conditioning* involves breaking the communication pathways along the loops by instantiating a select group of variables. *Stochastic simulation* involves assigning each variable a definite value and having each processor inspect the current state of its neighbors, compute the belief distribution of its host variable, and select one value at random from the computed distribution. Beliefs are then computed by recording the percentage of times that each processor selects a given value.

† Directed cycles, like those representing feedback in electronic circuits or econometric models, are not allowed in Bayesian networks and will not be discussed in this book.

The operation of the three schemes will be illustrated with a simple example borrowed from Spiegelhalter [1986], originally by Cooper [1984]:

> Metastatic cancer is a possible cause of a brain tumor and is also an explanation for increased total serum calcium. In turn, either of these could explain a patient falling into a coma. Severe headache is also possibly associated with a brain tumor.

Figure 4.23 shows the Bayesian network representing these relationships. As in the preceeding sections, we use uppercase letters to represent propositional variables and lowercase letters for their associated propositions. For example, $C \in \{1, 0\}$ represents the dichotomy between having a brain tumor and not having one. $+c$ stands for the assertion $C = 1$ or "Brain tumor is present," and $\neg c$ stands for the negation of $+c$, i.e., $C = 0$.

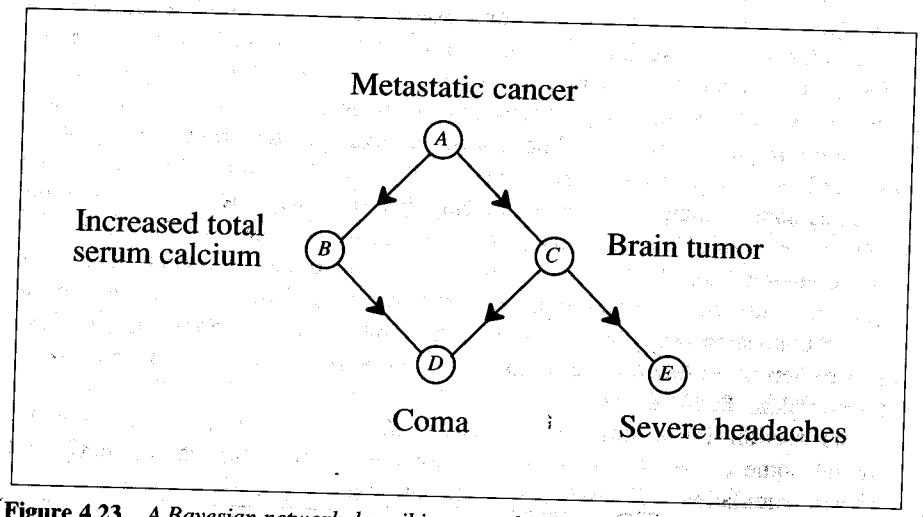

Figure 4.23. *A Bayesian network describing causal influences among five variables.*

Table 1 expresses the influences in terms of conditional probability distributions. Each variable is characterized by a *link matrix*, specifying the probability distribution of that variable given the state of its parents.† The root variable, having no parent, is characterized by its prior distribution.

† The probabilities are for illustration purposes only, and are not meant to realistically reflect current medical knowledge. Additionally, the variable "Coma" should be interpreted to mean "Lapsing occasionally into coma"; otherwise it would preclude headaches.

Table 1.

$P(a)$:	$P(+a) = .20$	
$P(b \mid a)$:	$P(+b \mid +a) = .80$	$P(+b \mid \neg a) = .20$
$P(c \mid a)$:	$P(+c \mid +a) = .20$	$P(+c \mid \neg a) = .05$
$P(d \mid b, c)$:	$P(+d \mid +b, +c) = .80$	$P(+d \mid \neg b, +c) = .80$
	$P(+d \mid +b, \neg c) = .80$	$P(+d \mid \neg b, \neg c) = .05$
$P(e \mid c)$:	$P(+e \mid +c) = .80$	$P(+e \mid \neg c) = .60$

Given this information, our task is to compute the posterior probability of every proposition in the system, given that a patient is suffering from severe headaches $(+e)$ but has not fallen into a coma $(\neg d)$, i.e., $\boldsymbol{e} = \{E = 1, D = 0\}$.

4.4.1 Clustering Methods

A straightforward way of handling the network of Figure 4.23 would be to collapse B and C into a single node representing the compound variable $Z = \{B, C\}$ with the values

$$z \in \{(+b, +c), (\neg b, +c), (+b, \neg c), (\neg b, \neg c)\}\ . \qquad \textbf{(4.65)}$$

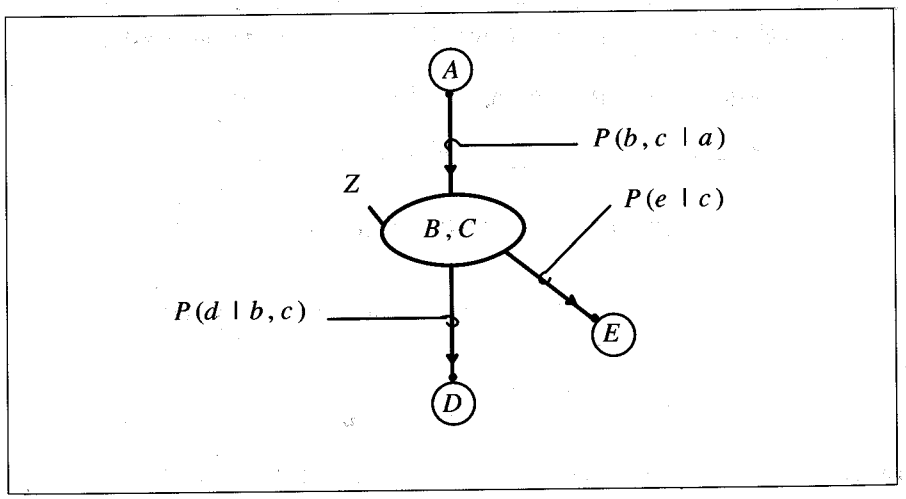

Figure 4.24. *Clustering B and C turns the network of Figure 4.23 into a tree.*

This results in the tree structure shown in Figure 4.24. Since the cardinality of variable Z is 4, the matrices on all three links must have either four rows or four

columns. This does not mean, however, that the enlarged matrices should be stored explicitly at their corresponding nodes; often they can be generated from the smaller matrices of Table 1. For example, the matrix $P(z \mid a)$ can be generated as the product

$$P(z \mid a) = P(b, c \mid a) = P(b \mid a)\, P(c \mid a)$$

$$= \begin{cases} (0.16, 0.04, 0.64, 0.16) & \text{if} \quad A = 1 \\ (0.01, 0.04, 0.19, 0.76) & \text{if} \quad A = 0, \end{cases}$$

and $P(e \mid z)$ can be generated by simply ignoring the b component of z:

$$P(e \mid z) = P(e \mid b, c) = P(e \mid c).$$

Still, the process of belief updating will proceed as though the four components of Z were independent entities.

The propagation of belief updates is conducted in the same fashion as with ordinary trees. The only differences are the increased dimensionality of the π and λ messages and the need to reaggregate $BEL(z)$ in case we wish to find $BEL(b)$ or $BEL(c)$.

EXAMPLE: As an illustration, let us calculate the belief distribution of all variables in Figure 4.23, given the evidence $e = \{\neg d, +e\}$. Initially, the π messages are given by

$$\pi_Z(a) = \pi(a) = (0.20, 0.80),$$

$$\pi_D(z) = \pi_E(z) = \pi(z) = \sum_a P(z \mid a)\, \pi(a)$$

$$= \Big[(0.20 \cdot 0.16 + 0.80 \cdot 0.01), (0.20 \cdot 0.04 + 0.80 \cdot 0.04),$$

$$(0.20 \cdot 0.64 + 0.80 \cdot 0.19)\ (0.20 \cdot 0.16 + 0.80 \cdot 0.76)\Big]$$

$$= \Big[0.04, 0.04, 0.28, 0.64\Big],$$

and the λ's are all unit vectors. Once D and E are instantiated, they generate

$$\lambda_D(z) = P(\neg d \mid z) = (0.20, 0.20, 0.20, 0.95),$$

$$\lambda_E(z) = P(+e \mid z) = (0.80, 0.80, 0.60, 0.60),$$

and these prompt node Z to compute

$$\lambda(z) = \lambda_D(z)\, \lambda_E(z) = (0.16, 0.16, 0.12, 0.57),$$

$$BEL(z) = \alpha\, \pi(z)\, \lambda(z)$$

$$= \alpha(0.04 \cdot 0.16, 0.04 \cdot 0.16, 0.28 \cdot 0.12, 0.64 \cdot 0.57)$$

$$= (0.0156, 0.0156, 0.0817, 0.887),$$

and generate

$$\pi_E(z) = \alpha\, \pi(z)\, \lambda_D(z) = \alpha(0.04 \cdot 0.20, 0.04 \cdot 0.20, 0.28 \cdot 0.20, 0.64 \cdot 0.95)$$

$$= (0.0118, 0.0118, 0.0823, 0.8941),$$

$$\pi_D(z) = \alpha\, \pi(z)\, \lambda_E(z) = \alpha(0.04 \cdot 0.80, 0.04 \cdot 0.80, 0.28 \cdot 0.60, 0.64 \cdot 0.60)$$

$$= (0.0519, 0.0519, 0.2727, 0.6233),$$

$$\lambda_Z(a) = \sum_z P(z \mid a)\, \lambda(z) = \Big[(0.16 \cdot 0.16 + 0.04 \cdot 0.16 + 0.64 \cdot 0.12 + 0.16 \cdot 0.57),$$

$$(0.01 \cdot 0.16 + 0.04 \cdot 0.16 + 0.19 \cdot 0.12 + 0.76 \cdot 0.57)\Big]$$

$$= (0.2, 0.464).$$

Node A now computes its belief distribution,

$$BEL(a) = \alpha\, \pi(a)\, \lambda_Z(a) = \alpha(0.20 \cdot 0.20, 0.80 \cdot 0.464) = (0.097, 0.903),$$

and the propagation process halts. To find $BEL(b)$ and $BEL(c)$ we write

$$BEL(b) = \sum_c BEL[Z = (b, c)] = \Big[(0.0156 + 0.0817), (0.0156 + 0.887)\Big]$$

$$= (0.097, 0.903),$$

$$BEL(c) = \sum_b BEL[Z = (b, c)] = \Big[(0.0156 + 0.0156), (0.0817 + 0.887)\Big]$$

$$= (0.031, 0.969).$$

SELECTING THE CLUSTERS

Every Bayesian network can be structured as a tree of clusters if we do not limit the size of the clusters. In the extreme case, we can lump together all non-leaf variables as one compound variable (see Figure 4.25*a*), which will yield a star

structure as in Figure 4.25*b*. This approach was taken by Cooper [1984] and Peng and Reggia [1986] in their medical diagnosis systems, where each state of the compound variable was regarded as a possible explanation of the findings obtained. Unfortunately, the exponential cardinality and structureless nature of the compound variable make it difficult to compute, much less explain, the beliefs accrued by individual hypotheses within this variable.

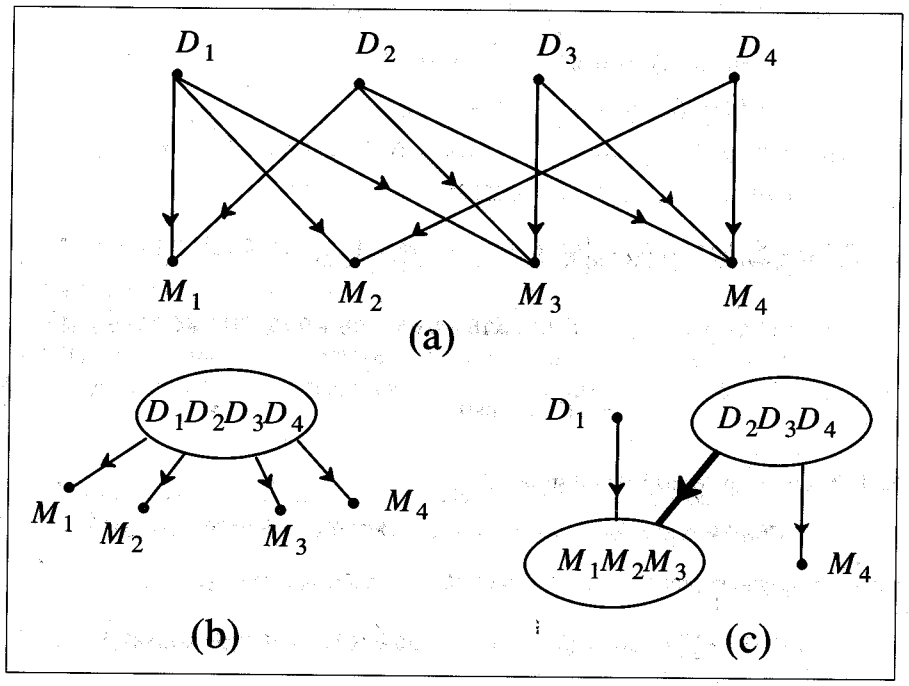

Figure 4.25. *Clustering the network (a) into a tree (b) and a polytree (c).*

A less dramatic clustering scheme is shown in Figure 4.25*c*. Here, the formation of the two clusters

$$Z_1 = \{D_2, D_3, D_4\} \quad \text{and} \quad Z_2 = \{M_1, M_2, M_3\}$$

renders the network of Figure 4.25*a* a polytree, where the propagation techniques of Section 4.3 are applicable. Ad hoc techniques are currently being used to choose a clustering structure; little work has been done in the area of finding the optimal structure given some criterion of performance.

One of the most popular methods of clustering is based on *join trees* (see Section 3.2.4). If the clusters are allowed to overlap each other until they cover all the links of the original network, then the interdependencies between any two clusters are mediated solely by the variables they share. If we insist that these

clusters continue to grow until their interdependencies form a tree structure, then the tree propagation scheme of Section 4.2 will be applicable.

We know from the discussion in Section 3.2.4 that if a probabilistic model P is decomposable with respect to a chordal graph G, the cliques of G can be arranged in a tree that is an I-map of P. This provides a simple, systematic method of forming clusters of variables for the purpose of propagating the impact of evidence in an arbitrary Bayesian network N_B:

1. Form the Markov network G of N_B by connecting all parents that share a common child and removing the arrows from the links (G is an I-map of N_B).
2. Form a chordal supergraph G' of G, using the graph-triangulation algorithm of Section 3.2.4 [Tarjan and Yannakakis 1984].
3. Identify the cliques of G' as compound variables, and connect them by links to form a join tree T.
4. Treat evidence nodes in N_B as dummy variables, transmitting λ-messages toward one of the clique nodes in T of which the evidence nodes are members.
5. Propagate the impact of these messages throughout T, and project the appropriate beliefs back to the individual variables of N_B.

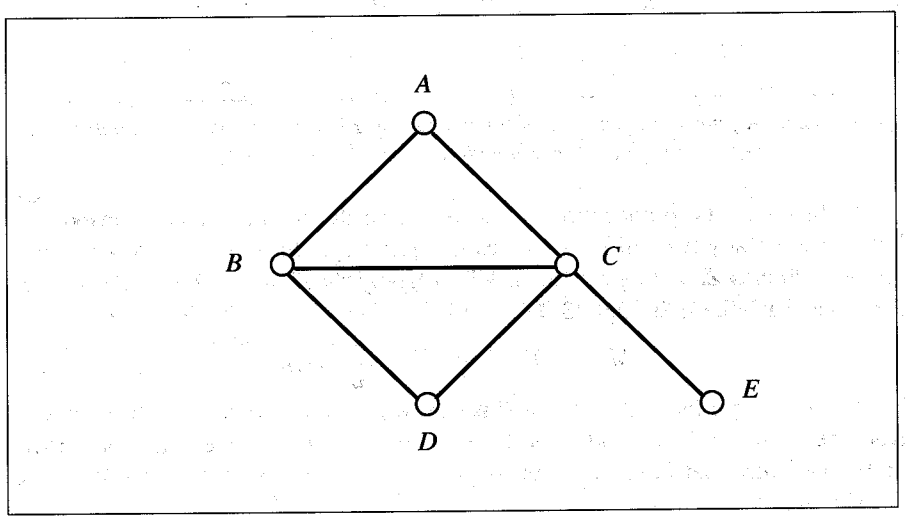

Figure 4.26. *The Markov network of the model in Figure 4.23.*

We illustrate this method on the model of Figure 4.23. The corresponding Markov network is formed by adding the link (B, C) between the two parents of D and removing all arrows (Figure 4.26). G is chordal and has three cliques:

$$Z_1 = \{A, B, C\}, \quad Z_2 = \{B, C, D\}, \quad Z_3 = \{C, E\}.$$

These cliques can be connected in two join tree structures (as in Figure 3.9), one of which is shown in Figure 4.27. The directionality of the arrows was chosen to match the directionality of the original network, Figure 4.23. The evidence variables D and E appear in Z_2 and Z_3, respectively, so dummy links are formed toward these two cliques, carrying λ-messages that exclude those states of Z_2 and Z_3 that are incompatible with $D = 0$ and $E = 1$.

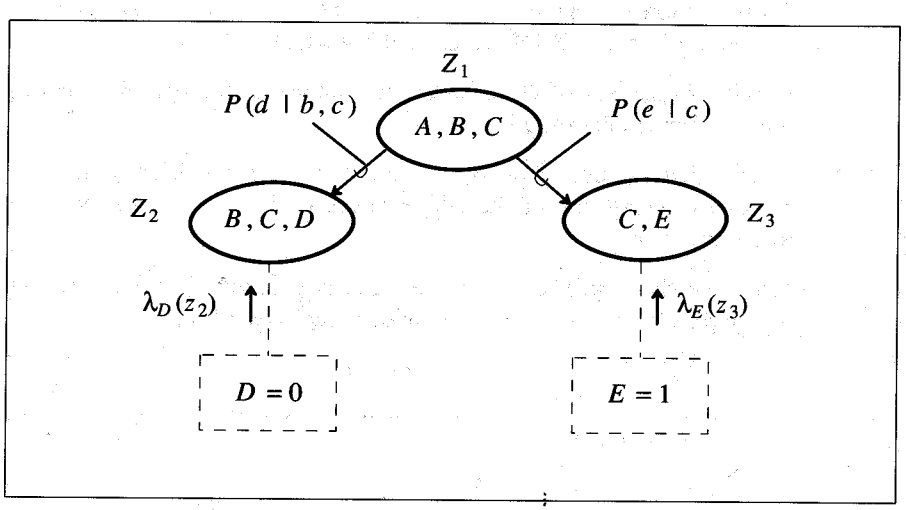

Figure 4.27. *A join tree clustering of the network in Figure 4.26, with two dummy nodes representing the observed findings.*

To facilitate the propagation we need to find the matrices that correspond to links between any two adjacent cliques in a join tree. Since the interdependencies between cliques Z_i and Z_j are mediated solely by the shared variable $Z_i \cap Z_j$, we have (see Eqs. (3.24) through (3.26))

$$M_{z_i|z_j} = P(z_i \mid z_j) = P(z_i \mid z_i \cap z_j).$$

Thus, if Z_j is a parent of Z_i in T, the link between the two cliques can be computed from the conditional probability of the group of variables unique to Z_i, conditioned on the variables that Z_i shares with its parent Z_j. In our example, since Z_2 shares $\{B, C\}$ with Z_1, we have

$$M_{z_2|z_1} = P(z_2 \mid z_1) = P(b, c, d \mid b, c) = P(d \mid b, c). \qquad \textbf{(4.66)}$$

Although the $M_{z_2|z_1}$ matrix should, in principle, measure 8 by 8, the requirement that the values assigned to B and C by Z_2 be identical to those assigned by Z_1 renders $M_{z_1|z_2}$ fully characterized by the four parameters in $P(d \mid b, c)$, as in Table 1. The computations will proceed as though the link matrix measured 8 by 8 and the messages traversing this link were eight-dimensional.

Let us demonstrate now how the evidence $\{\neg d, +e\}$ propagates its impact through the join tree of Figure 4.27. Before this evidence is observed the probabilities and π-messages are given by

$$\pi(z_1) = \pi_{Z_2}(z_1) = \pi_{Z_3}(z_1) = P(z_1) = P(a, b, c) = P(b \mid a)\, P(c \mid a)\, P(a). \qquad \textbf{(4.67)}$$

The λ-messages are all unit vectors. Eq. (4.67) stands for the eight-component vector shown in the $\pi(z_1)$ column of Table 2.

Table 2.

a,	b,	c	$P(b \mid a)$	$P(c \mid a)$	$Pa)$	$\pi(z_1)$
1	1	1	0.80	0.20	0.20	0.032
1	1	0	0.80	(1 - 0.20)	0.20	0.128
1	0	1	(1 - 0.80)	0.20	0.20	0.008
1	0	0	(1 - 0.80)	(1 - 0.20)	0.20	0.032
0	1	1	0.20	0.05	(1 - 0.20)	0.008
0	1	0	0.20	(1 - 0.05)	(1 - 0.20)	0.152
0	0	1	(1 - 0.20)	(0.05)	(1 - 0.20)	0.032
0	0	0	(1 - 0.20)	(1 - 0.05)	(1 - 0.20)	0.608

In practice, these components can be generated upon demand, using the formula of Eq. (4.67), from the five parameters given in Table 1. The same applies to the prior probabilities of the other cliques:

$$\pi(z_2) = \sum_{Z_1} P(z_2 \mid z_1)\, P(z_1) = \sum_{a} P(d \mid b, c)\, P(a, b, c) = P(d \mid b, c)\, P(b, c),$$

$$\pi(z_3) = \sum_{Z_1} P(z_3 \mid z_1)\, P(z_1) = \sum_{a,b} P(e \mid c)\, P(a, b, c) = P(e \mid c)\, P(c).$$

$P(b, c)$ and $P(c)$ can be computed by summing over the appropriate terms of $\pi(z_1)$.

When the evidence $D = 0$ arrives, it sets up a λ-message for Z_2 which eliminates all states of Z_2 incompatible with $D = 0$, i.e.,

$$\lambda_D(z_2) = \begin{cases} 0 & \text{if } z_2 = (b, c, +d) \\ 1 & \text{if } z_2 = (b, c, \neg d). \end{cases}$$

Similarly, $E = 1$ generates

$$\lambda_E(z_3) = \begin{cases} 0 & \text{if } z_3 = (c, \neg e) \\ 1 & \text{if } z_3 = (c, +e). \end{cases}$$

These messages prompt Z_2 and Z_3 to generate corresponding λ messages for Z_1,

$$\lambda_{Z_2}(z_1) = M_{z_2|z_1} \bullet \lambda_D(z_2) = P(\neg d \mid b, c),$$

$$\lambda_{Z_3}(z_1) = M_{z_3|z_1} \bullet \lambda_E(z_3) = P(+e \mid c),$$

permitting Z_1 to compute its belief distribution:

$$BEL(z_1) = \alpha \lambda_{Z_2}(z_1) \lambda_{Z_3}(z_1) \pi(z_1)$$

$$= \alpha P(\neg d \mid b, c) P(+e \mid c) P(b \mid a) P(c \mid a) P(a).$$

The computation yields an eight-dimensional vector from which $BEL(a)$, $BEL(b)$, and $BEL(c)$ can be computed by the appropriate summations.

The close similarity between this computation and the one conducted using $Z = \{B, C\}$ as the only cluster should not be surprising. Although the new clusters have more dimensions, most of the computations within clusters are still conducted using the set $\{B, C\}$ as a mediator for A, D, and E. Taking advantage of this fact, Lauritzen and Spiegelhalter [1988] have suggested a variation of the join tree method whereby the compound variables used in the propagation phase are the intersection sets between any two adjacent cliques in the join tree.

4.4.2 The Method of Conditioning (Reasoning by Assumptions)

Conditioning is based on our ability to change the connectivity of a network and render it singly connected by instantiating a selected group of variables. In Figure 4.23, for example, instantiating A to any value would block the pathway B—A—C and would render the rest of the network singly connected, so that the propagation techniques of Section 4.3 would be applicable. Thus, if we wish to propagate the impact of an observed fact, say $E = 1$, to the entire network, we first assume $A = 0$ (as in Figure 4.28*a*), propagate the impact of $+e$ to the variables B, C, and D, repeat the propagation under the assumption $A = 1$ (as in Figure 4.28*b*), and

finally, average the two results weighted by the posterior probabilities $P(A = 1 \mid E = 1)$ and $P(A = 0 \mid E = 1)$. We can also execute the propagation in parallel by letting each node receive, compute, and transmit two sets of parameters, one for each value of the conditioning variable A. Conditioning provides a working solution in many cases, but like clustering, if the network is highly connected it may suffer from a combinatorial explosion—the message size grows exponentially with the number of nodes required for breaking up the loops in the network.

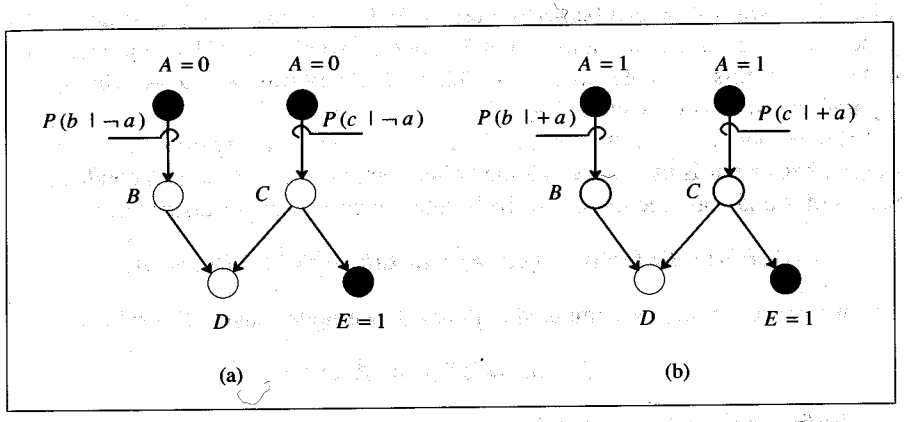

Figure 4.28. *The multiply connected network of Figure 4.23 is decomposed into two polytrees corresponding to the two instantiations of A.*

The use of conditioning to facilitate propagation is not foreign to human reasoning. When we find it hard to estimate the likelihood of a given outcome, we often make hypothetical *assumptions* that render the estimation simpler, and then negate the assumptions to see if the results vary substantially. One pervasive pattern of plausible reasoning is the maxim that if two diametrically opposed assumptions impart different degrees of confidence onto a proposition Q, then the unconditional degree of confidence merited by Q should be somewhere between them (see Section 1.4.2). The terms *hypothetical reasoning*, *assumption-based reasoning*, *reasoning by cases*, and *envisioning* refer to the same basic mechanism of selecting a key variable, binding it to some values, deriving the consequences of each binding separately, and integrating the consequences.

Conditioning draws its legitimacy from the ever-faithful rule of total probabilities, which for any three variables, X, Y, and Z, permits us to write

$$P(y \mid z) = \sum_x P(y \mid x, z) P(x \mid z).$$

In our example, if we wish to compute the belief distribution associated with the variable B given the evidence $E = e$, we can choose A as the conditioning variable and write

$$BEL(b) = P(b \mid e) = \sum_a P(b \mid a, e)\, P(a \mid e). \tag{4.68}$$

The first term in the summation stands for the belief distribution of B that one would calculate by propagating the impact of $E = e$ through a network clamped at $A = a$. Since this clamping of A renders the network singly connected (see Figure 4.28), the computation can be performed swiftly. Note that C would be an equally good choice as a conditioning variable, but D would be a bad choice, since instantiating this variable would not block the pathway A—D—C (by the d-separation criterion of Section 3.4).

The second term in the summation, $P(a \mid e)$, can be regarded as a *mixing weight*, because it is used to weigh the beliefs obtained under the two conditioning values of A and combine them additively into the correct belief distribution:

$$BEL(b) = BEL(b \mid a)\, P(+a \mid e) + BEL(b \mid \neg a) \left[1 - P(+a \mid e)\right].$$

This weight can easily be computed at the evidence node E using Bayes' Rule:

$$P(a \mid e) = \alpha\, P(e \mid a)\, P(a).$$

The first term on the right is the conditional probability associated with E prior to the observation; it can be obtained by propagating the impact of $A = a$ through a singly connected network onto E. The second term is simply the prior probability of A, which can be passed to every node of the network before any evidence arrives.

In general, if the evidence comprises several instantiated nodes E^1, E^2..., then the overall mixing weight $P(a \mid e^1, e^2, \ldots)$ can be computed recursively; every instantiated node E^i, in its turn, computes the new mixing weight $P(a \mid e^1, \ldots, e^{i-1}, e^i)$ from the old one $P(a \mid e^1, \ldots, e^{i-1})$ and passes it along to all the other variables. This computation is, again, best done by the product

$$P(a \mid e^1, \ldots, e^{i-1}, e^i) = \alpha\, P(e^i \mid a, e^1, \ldots, e^{i-1})\, P(a \mid e^1, \ldots, e^{i-1}),$$

because the two terms on the right are available to E^i at the moment of its instantiation: the first stands for the current $BEL(e^i)$ distributions (corresponding to the two conditioning values of A), and the second is the previous mixing weight.

EXAMPLE: To show how conditioning works, let us return to the problem of Figure 4.23 and compute the belief distributions of A, B, and C given the evidence $\{\neg d, +e\}$. Initially, we compute the belief distribution for B, C, D, and E under the two conditions

$A = 1$ and $A = 0$ by propagating these two values down the polytrees of Figure 4.28. Denoting the two conditions above by superscripts 1 and 0, we write

$$\pi^1(b) = P(b \mid a) = (0.80, 0.20),$$
$$\pi^1(c) = P(c \mid a) = (0.20, 0.80),$$

$$BEL^1(+d) = \sum_{b,c} P(+d \mid b, c)\, \pi^1(b)\, \pi^1(c)$$
$$= \big[0.80 \cdot 0.80 \cdot 0.20 + 0.80(1 - 0.80)\, 0.20 + 0.80 \cdot 0.80(1 - 0.20)$$
$$+ 0.05(1 - 0.80)\, (1 - 0.20)\big]$$
$$= 0.68,$$

$$BEL^1(+e) = \pi^1(+e) = \sum_c P(+e \mid c)\, \pi^1(c) = 0.80 \cdot 0.20 + 0.60 \cdot 0.80 = 0.64,$$
$$\pi^0(b) = P(b \mid \neg a) = (0.20, 0.80),$$
$$\pi^0(c) = P(c \mid \neg a) = (0.05, 0.95),$$

$$BEL^0(+d) = \pi^0(+d) = \sum_{b,c} P(+d \mid b, c)\, \pi^0(b)\, \pi^0(c)$$
$$= \big[0.80 \cdot 0.20 \cdot 0.05 + 0.80 \cdot (1 - 0.20)\, 0.05 + 0.80 \cdot 0.20(1 - 0.05)$$
$$+ 0.05(1 - 0.20)\, (1 - 0.05)\big]$$
$$= 0.23,$$

$$BEL^0(+e) = \pi^0(+e) = \sum_c P(+e \mid c)\, \pi^0(c) = 0.80 \cdot 0.05 + 0.60 \cdot 0.95 = 0.61.$$

This initial message profile is shown in Figure 4.29. Each node stores its two belief distributions BEL^0 and BEL^1, together with the initial mixing weight

$$w = (w^1, w^0) = P(a) = (0.20, 0.80).$$

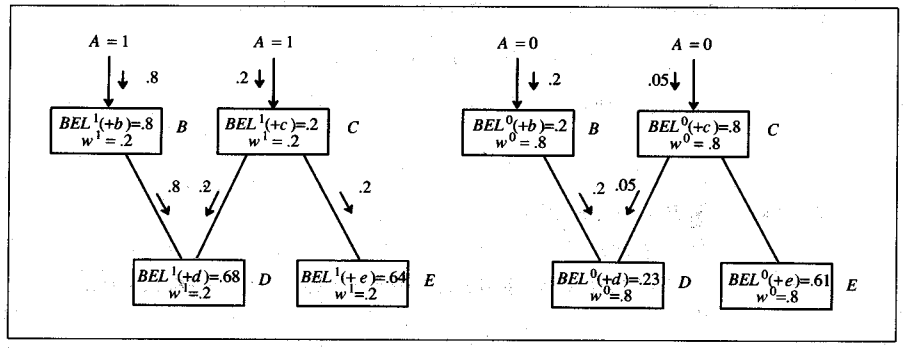

Figure 4.29. *Initial beliefs, messages, and weights under two assumptions.*

Now imagine the evidence $E = 1$ is observed. E computes and sends to all other nodes the new mixing weight:

$$\begin{aligned} w_E &= P(a \mid +e) = \alpha P(+e \mid a) P(a) \\ &= \alpha \left[BEL^1(+e)\, w^1, BEL^0(+e)\, w^0 \right] \\ &= \alpha \left[0.64 \cdot 0.20, 0.61 \cdot 0.80 \right] \\ &= (0.208, 0.792). \end{aligned}$$

Simultaneously, E posts the messages $\lambda_E^1(c)$ and $\lambda_E^0(c)$ for C, where (since $P(e \mid c,a) = P(e \mid c)$)

$$\lambda_E^1(c) = \lambda_E^0(c) = P(+e \mid c) = (0.80, 0.60).$$

Node C now computes two $\pi_D(c)$ messages for D, $\pi_D^1(c)$ and $\pi_D^0(c)$, corresponding to the two conditioning values of A:

$$\begin{aligned} \pi_D^1(c) = BEL^1(c) &= \alpha^1 \pi^1(c)\, \lambda_E(c) = \alpha^1 (0.20, 0.80)\, (0.80, 0.60) \\ &= (0.25, 0.75), \end{aligned}$$

$$\begin{aligned} \pi_D^0(c) = BEL^0(c) &= \alpha^0 \pi^0(c)\, \lambda_E(c) = \alpha^0 (0.05, 0.95)\, (0.80, 0.60) \\ &= (0.066, 0.934). \end{aligned}$$

This results in the belief distributions

$$BEL^1(d) = \sum_{b,c} P(d \mid b, c)\, \pi^1(b)\, \pi_D^1(c) = (0.6875, 0.3125),$$

$$BEL^0(d) = \sum_{b,c} P(d \mid b, c)\, \pi^0(b)\, \pi_D^0(c) = (0.24, 0.76).$$

At this point, the propagation of these two sets of messages halts, because as long as D is an anticipatory node, the pathway C—D—B is blocked at D (see Figure 4.30).

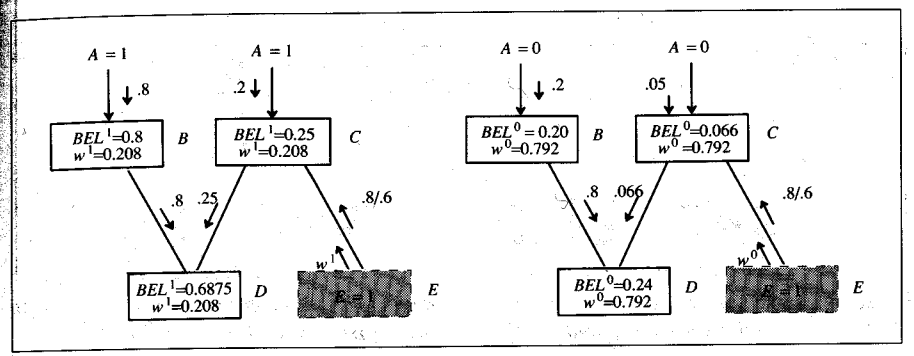

Figure 4.30. *Updated beliefs, messages, and weights after observing $E = 1$.*

The arrival of the next piece of evidence, $D = 0$, prompts D to compute the new mixing weight $w_{E,D}$ and then initiate a new message-passing process by generating $\lambda_D(b)$ and $\lambda_D(c)$:

$$\begin{aligned} w_{E,D} &= P(a \mid +e, \neg d) = \alpha P(\neg d \mid a, +e)\, P(a \mid +e) \\ &= \alpha \left[BEL^1(\neg d)\, w_E^1, BEL^0(\neg d)\, w_E^0 \right] \\ &= \alpha (0.3125 \cdot 0.208, 0.76 \cdot 0.792) = (0.0975, 0.9025), \end{aligned}$$

$$\begin{aligned} \lambda_D^1(c) = \sum_b P(\neg d \mid b, c)\, \pi_D^1(b) &= \left[(0.2 \cdot 0.8 + 0.2 \cdot 0.2), (0.2 \cdot 0.8 + 0.95 \cdot 0.2) \right] \\ &= (0.20, 0.35) \end{aligned}$$

$$\begin{aligned} \lambda_D^0(c) = \sum_b P(\neg d \mid b, c)\, \pi_D^0(b) &= \left[(0.2 \cdot 0.2 + 0.2 \cdot 0.8), (0.2 \cdot 0.2 + 0.95 \cdot 0.8) \right] \\ &= (0.20, 0.8), \end{aligned}$$

$$\begin{aligned} \lambda_D^1(b) = \sum_c P(\neg d \mid b, c)\, \pi_D^1(c) &= \left[(0.2 \cdot 0.25 + 0.2 \cdot 0.75), (0.2 \cdot 0.25 + 0.95 \cdot 0.75) \right] \\ &= (0.2, 0.76) \end{aligned}$$

$$\begin{aligned} \lambda_D^0(b) = \sum_c P(\neg d \mid b, c)\, \pi_D^0(c) &= \left[(0.2 \cdot 0.066 + 0.2 \cdot 0.934), (0.2 \cdot 0.066 + 0.95 \cdot 0.934) \right] \\ &= (0.2, 0.9). \end{aligned}$$

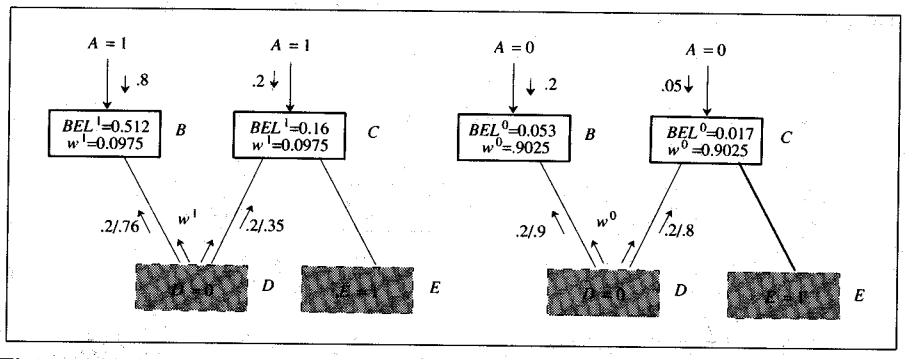

Figure 4.31. *Updated beliefs, messages, and weights after observing $E = 1$ and $D = 0$. Beliefs are computed by the combination $BEL = w^1\, BEL^1 + w^0\, BEL^0$.*

At this point, all belief distributions can be computed at their corresponding nodes, as in Figure 4.31:

$$BEL(b) = w^1_{E,D}\, BEL^1(b) + w^0_{E,D}\, BEL^0(b),$$
$$BEL(c) = w^1_{E,D}\, BEL^1(c) + w^0_{E,D}\, BEL^0(c),$$

where

$$BEL^1(b) = \alpha^1\, \pi^1(b)\, \lambda^1_D(b) = \alpha^1(0.8 \cdot 0.2, 0.2 \cdot 0.76) = (0.512, 0.488),$$
$$BEL^0(b) = \alpha^0\, \pi^0(b)\, \lambda^0_D(b) = \alpha^0(0.2 \cdot 0.2, 0.8 \cdot 0.9) = (0.053, 0.947),$$
$$BEL^1(c) = \alpha^1\, \pi^1(c)\, \lambda^1_D(c)\, \lambda_E(c) = \alpha^1(0.2 \cdot 0.2 \cdot 0.80, 0.8 \cdot 0.35 \cdot 0.60)$$
$$= (0.16, 0.84),$$
$$BEL^0(c) = \alpha^0\, \pi^0(c)\, \lambda^0_D(c)\, \lambda_E(c) = \alpha^0(0.05 \cdot 0.2 \cdot 0.80, 0.95 \cdot 0.8 \cdot 0.60)$$
$$= (0.017, 0.983).$$

These yield

$$BEL(b) = (0.096, 0.904),$$
$$BEL(c) = (0.031, 0.964).$$

$BEL(a)$, of course, is equal to the current mixing weight $w_{E,D} = (0.0975, 0.9025)$.

4.4.3 Stochastic Simulation

Stochastic simulation is a method of computing probabilities by counting how frequently events occur in a series of simulation runs. If a causal model of a domain is available, the model can be used to generate random samples of

hypothetical scenarios that are likely to develop in the domain. The probability of any event or combination of events can then be computed by counting the percentage of samples in which the event is true.

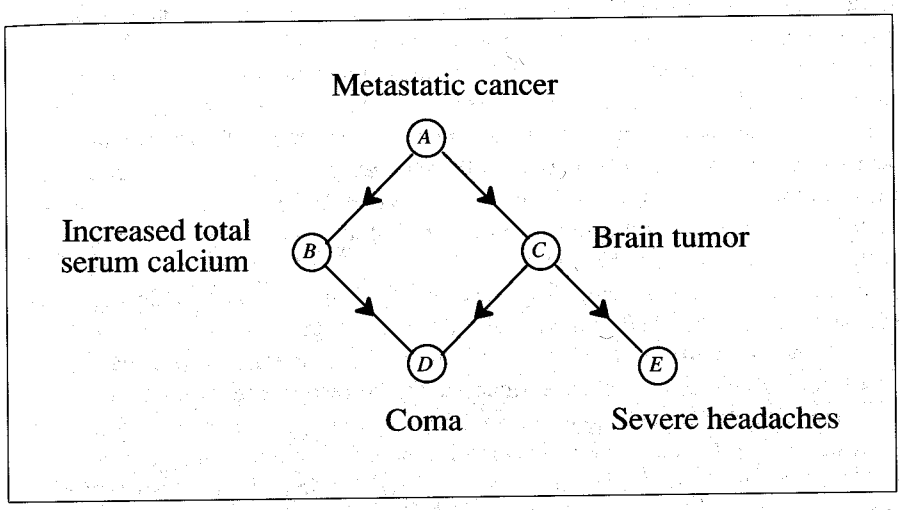

Figure 4.32. *The Bayesian network used to demonstrate stochastic simulation (same as in Figure 4.23).*

For example, in the causal model of Figure 4.32, we can generate hypothetical samples of patients by the following procedure: We draw a random value a_1 for A, using the probability $P(a)$. Given a_1, we draw random values b_1 and c_1 for the variables B and C, using the probabilities $P(b \mid a_1)$ and $P(c \mid a_1)$, respectively. Given b_1 and c_1, we draw random values d_1 and e_1 for D and E, using $P(d \mid b_1, c_1)$ and $P(e \mid c_1)$, respectively. The combination of values $(a_1, b_1, c_1, d_1, e_1)$ represents one sample of a patient scenario. The process now repeats from A down to D and E, each run generating a quintuple that represents one patient.

Stochastic simulation shows considerable potential as a probabilistic inference engine that combines evidence correctly but is computationally tractable. Unlike numerical schemes, the computational effort is unaffected by the presence of dependencies within the causal model; simulating the occurrence of an event given the states of its causes requires the same computational effort regardless of whether the causes are correlated. In our example above, simulating the event D given the states of events B and C was straightforward, even though B and C are correlated (via A). Thus, the presence of loops in the network does not affect the computation.

Stochastic simulation carries a special appeal for AI researchers in that it develops probabilistic reasoning as a direct extension of deterministic logical