Knowledge-Based Probabilistic Reasoning from Expert Systems to Graphical Models

By

George F. Luger & Chayan Chakrabarti
{luger | cc} @cs.unm.edu
Department of Computer Science
University of New Mexico
Albuquerque NM 87131

An important research enterprise for the Artificial Intelligence community since the 1970s has been the design of expert or "knowledge-based" systems. These programs used explicitly encoded human knowledge, often in the form of a production rule system, to solve problems in the areas of diagnostics and prognostics. The earliest research/development program in expert systems was created by Professor Edward Feigenbaum at Stanford University (Buchanan and Shortliff 1984). Because the expert system often addresses problems that are imprecise and not fully proposed, with data sets that are often inexact and unclear, the role of various forms of probabilistic support for reasoning is important.

The 1990s saw radical new approaches to the design of automated reasoning/diagnostic systems. With the creation of graphical models, the explicit pieces of human knowledge (of the expert system) were encoded into causal networks, sometimes referred to as Bayesian belief networks (BBNs). The reasoning supporting these networks, based on two simplifying assumptions (that reasoning could not be cyclic and that the causality supporting a child state would be expressed in the links between it and its parent states) made BBN reasoning quite manageable computationally. In recent years the use of graphical models has replaced the traditional expert system, especially in situations where reasoning was diagnostic and prognostic, i.e., extending from concrete situations to the best explanations for their occurrence. This type reasoning is often termed *abductive*.

In this chapter we first (Section 1) present the technology supporting the traditional knowledge-based expert system, including the production system for reasoning with rules. Next (Section 2), we discuss Bayesian inference, and the adoption of simplifying techniques such as the Stanford Certainty Factor Algebra. We then (Section 3) introduce graphical models, including the assumptions supporting the use of Bayesian belief networks (BBN), and present an example of BBN reasoning. We conclude (Section 4) with a brief introduction of a next generation system for diagnostic reasoning with more expressive forms of the BBN.

Section 1: Expert Systems

We begin with the presentation of some of the traditional application areas of the rule based technology. We then describe the software architecture supporting its development.

1.1 Introduction of the Traditional Expert System

The rationale behind "knowledge-based" problem solvers was that human experts knew a lot about their area of expertise. Expert systems designers acquire this knowledge from human experts and then program the system to emulate th4e human expert's methodology. These human experts also augment the system's knowledge with tricks, shortcuts and heuristics that they have gained from experience. These heuristics can also be encoded using probabilistic methods. Expert systems are built to solve a wide range of problems in domains such as medicine, mathematics, engineering, chemistry, geology, computer science, business, law, defense, and education. The range of problem categories can be summarized as follows

(Waterman 1986).

Interpretation—forming high-level conclusions from collections of raw data.
Prediction—projecting probable consequences of given situations.
Diagnosis—determining the cause of malfunctions in complex situations based on observable symptoms.
Design—finding a configuration of system components that meets performance goals while satisfying a set of design constraints.
Planning—devising a sequence of actions that will achieve a set of goals given certain starting conditions and run-time constraints.
Monitoring—comparing a system's observed behavior to its expected behavior.
Instruction—assisting in the education process in technical domains.
Control—governing the behavior of a complex environment.

Generally, expert systems programs tend to support the iterative development methodology. This requires that programs be easily prototyped, tested and changed. Easy modification of the knowledge base is vital for successful expert system design. Also, it is important that the expert system can display all the intermediate problem solving steps and can justify choices and decisions. These explanations are important for a human expert, such as a doctor or an engineer, if he is to accept the system's recommendations.

1.2 The Design of Rule-Based Expert Systems

Figure 1.1 shows the modules that make up a typical expert system. The user interface, often graphical, hides much of the complexity of the system. All the knowledge of a problem domain is encoded in the knowledge-base, which is the heart of the system. Most often, this consists of if... then... rules. The inference engine applies the knowledge to the solution of actual problems. This can be done using a production system (Section 1.3). The knowledge base and inference engine are separated for several reasons:

1. If ... then... rules are a very natural way to represent human problem-solving skills.

2. System designers can focus on capturing problem-solving knowledge without worrying about implementation details.

3. Changes can be made in one part of the knowledge-base without affecting others.

4. The same inference engine can be plugged into different expert systems and work on different knowledge-bases.

The knowledge-base is augmented with case-specific data, which contains information relevant to the case under consideration. The explanation subsystem allows the program to explain its reasoning to the user.

1.3 The Production System in Expert System Problem Solving

The *production system* is a model of computation that has proved particularly important in artificial intelligence, both for implementing search algorithms and for modeling human problem solving. A production system provides pattern-directed control of a problem-solving process and consists of a set of *production rules*, a *working memory*, and a *recognize–act* control cycle.

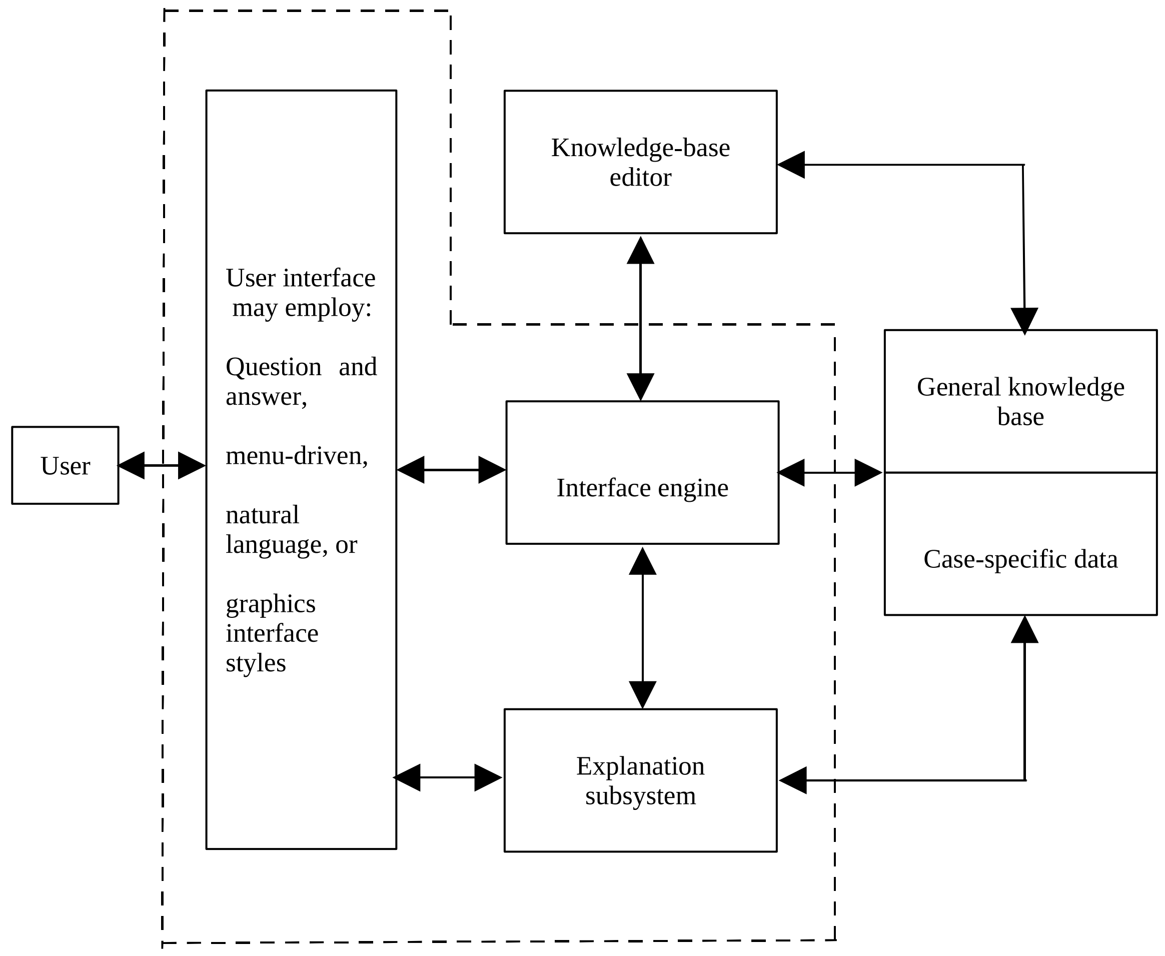

Figure 1.1 Architecture of a typical expert system for a particular problem domain, adapted from Luger (2005).

A *production system* may be defined:

The set of production rules. These are often simply called *productions*. A production is a *condition → action* pair and defines a single chunk of problem-solving knowledge. Both the condition part and the goal part of each rule is a pattern that determines when that rule may be applied to a problem instance.

Working memory contains a description of the *current state of the world* in a reasoning process. This description is a pattern that, in *data-driven reasoning* is matched against the condition part of the set of productions to select appropriate problem-solving actions and in *goal-driven reasoning* is matched against the current goal or sub-goal being explored.

The recognize–act cycle. The control structure for a production system is simple: *working memory* is initialized with the beginning and goal problem descriptions. The current state of the problem-solving is maintained as a set of patterns in working memory. These patterns are matched against the conditions or actions of the production rules; this produces a subset of the production rules, called the *conflict set*, whose conditions (or goals) match the patterns in working memory. After a selected production rule is fired, the control cycle repeats with the modified working memory. The process terminates when the contents of working memory do not match any rules or the problem is solved.

Conflict resolution is a set of heuristics that is used to choose a rule from the conflict set for firing. Conflict resolution strategies may be simple, such as selecting the first rule whose condition or action matches the state of the world, or may involve complex rule selection heuristics (Luger 2005, Chapter 6; Forgy 1982).

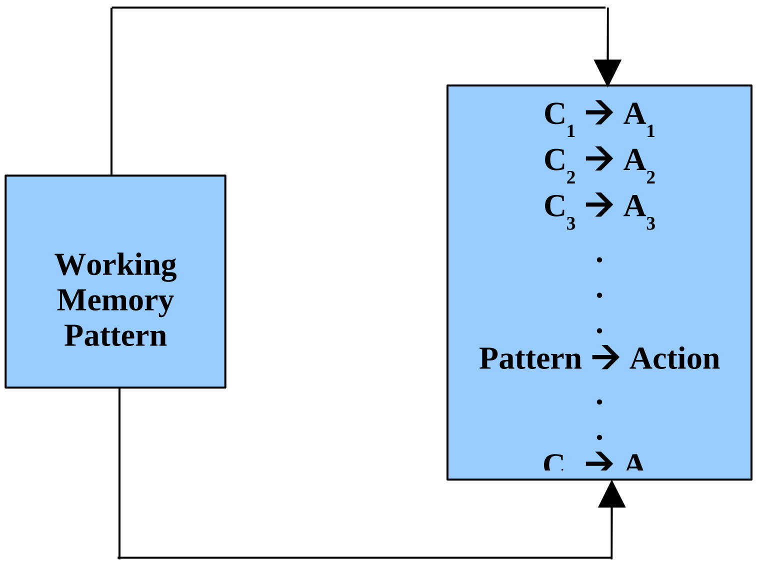

Figure 1.2 A production system. Control loops until working memory pattern no longer matches the conditions of any productions, adapted from Luger (2005).

The architecture of rule-based expert systems may be best understood in terms of the production system model for problem solving. The parallel between the two is more than an analogy: the production system was the intellectual precursor of modern expert system architectures, where application of production rules leads to refinements of understanding of a particular problem situation. When Newell and Simon first developed the production system, their goal was to model human performance in problem solving (Newell and Simon 1976).

If we regard the expert system architecture in Figure 1.2 as a production system, the domain-specific knowledge base is the set of production rules. In a rule-based system, these condition action pairs are represented as *if… then…* rules, with the premises of the rules, the if portion, corresponding to the condition, and the conclusion, the then portion, corresponding to the action, In data-driven reasoning, when the condition is satisfied, the expert system takes the action of asserting the conclusion as true. In goal-driven reasoning, the then portion of the rule is matched to see what conditions (sub-goals) must be supported for the goal to be true. Case-specific data can be kept in the working memory. The inference engine implements the recognize-act cycle of the production system that may be either data-driven or goal-driven.

Many problem domains seem to lend themselves more naturally to forward search. In an interpretation problem, for example, most of the data for the problem are initially given and it is often difficult to formulate an hypotheses or goal. This suggests a forward reasoning process in which the facts are placed in working memory, conditions of rules matching those facts are matched, and the system searches for an interpretation.

In a goal-driven expert system, the goal expression is initially placed in working memory. The system matches rule conclusions with the goal, selecting one rule and placing its premises in the working memory. This corresponds to a decomposition of the problem's goal into simpler sub-goals. The process continues in the next iteration of the production system, with these sub-goals becoming the new goals to match against rule conclusions. The system thus works back from the original goal until all the sub-goals in working memory are known to be true, indicating that the hypothesis has been verified. Thus, backward search in an expert system corresponds roughly to the process of hypothesis testing in human problem solving,

In an expert system, sub-goals can be solved (when the production rules offer no matches) by asking the system's user for information. Further, some expert systems allow the designer to specify which sub-goals are to be solved by asking the user. Others simply ask the user about any sub-goals that fail to match rules in the knowledge base; i.e., if the program cannot infer the truth of a sub-goal, it asks the user.

Section 2: Knowledge-Based Reasoning in Conditions of Uncertainty

We next present the technology supporting uncertain reasoning in expert systems. Although a few early systems used Bayesian approaches these by and large proved unsuitable to the task. We discuss these issues and then present a popular alternative, the Stanford certainty factor algebra.

2.1 A Brief Introduction to the Bayesian Approach

To this point in our presentation we have ignored an important component of expert system work. That is, in many applications the *if… then…* rules of the system are NOT always certain "if and only if" or cause/effect relationships. That is, many of the rules express "usually", "it is likely that", "often", or "sometimes" relations between their conditions and the actions.

This is not always the case, of course. The rules in many expert systems express direct cause and effect; for example, in software to configure a computer hardware system there will be no uncertainty as to whether a certain disk and bus can be linked. Furthermore, in working with algebraic expert systems there is no uncertainty involved in relationships, for example, between a function and its derivative.

Most expert systems, however, especially those reasoning from effects back to causes, do have some uncertainty in their condition action relationships. Medical systems, and MYCIN is certainly the prototype (Buchanan and Shortliff 1984) of this genre, reason from patient's symptoms back to possible causes (explanations) for these symptoms. In fact, the vast majority of diagnostic expert systems reason back from the facts of situations to their probable causes. The vast majority of these relationships are inherently uncertain!

A further issue that impacts uncertain reasoning is the vagueness and lack of clarity that is often found in data sets. When a rule asks if a patient's temperature is "high" what precisely does that mean? Even if the rule is rewritten using a range of explicit temperatures, certainly some will be "higher" than others. There will also be some variation within the testing devices that are used to obtain patient's temperatures. Finally, in many diagnostic situations key pieces of data may in fact be missing. In all these situations, the expert system must still track what information it has and attempt to come up with the best possible explanation.

Bayes' theorem (1763) is a mathematically sound methodology for interpreting uncertain relationships, such as those just mentioned. Further, Bayes' equation relates the a priori (what we know about and/or have been trained to "see") with the a posteriori (what we are currently observing). We will not describe Bayes' work in detail here – this has been done adequately in other places in this handbook. Although Bayes' theorem is often seen as natural for expert system work, and several systems have used it, most famously PROSPECTOR (Duda et al. 1979), we make the larger point that Bayes' theorem is NOT used in the development of the vast majority of expert systems. We then describe the most common alternative for uncertain reasoning in expert systems, the Stanford Certainty Factor Algebra.

There are several major reasons for not using Bayes' theorem to address the uncertainty requirements in rule based expert systems. The first reason, is the task of collecting all the necessary probability measures. This is especially difficult because most expert systems are dynamic: if the designer wants to add new "knowledge" to the system, she simply adds one or more new production rules. This, of course, requires the strict Bayesian to readjust many of the probability measures already in the system to accommodate this new situation. Second, many of the uncertainties of the knowledge-based system are not probabilistic and/or difficult to assign an appropriate probability measure. Finally, we must assume in our application domain that the pieces of evidence for a particular hypothesis are independent. In many important problem domains this assumption of independence cannot be justified.

It is interesting to note, however, that many situations that violate this assumption (that the individual pieces of evidence partition the evidence set) behave quite well! Using this partition assumption, even in situations where it is not justified, is called using *naive Bayes* or a *Bayes classifier* (Luger 2005, Chapter 5).

In general, we use Bayes' theorem to determine the probability of some hypothesis h_i given a set of evidence E, or $p(h_i \mid E)$. In this situation, Bayes requires that we obtain the values of $p(E \mid h_i)$ and $p(h_i)$. These numbers are often much more easily obtainable, compared to obtaining the values for $p(h_i \mid E)$ directly. For example, because the population is smaller, it is much easier to determine the number of meningitis patients who have headaches than it is to determine the percentage of headache sufferers that have meningitis. Even more importantly, for the simple case of a single disease and a single symptom, not very many numbers are needed. Troubles begin, however, when we consider multiple hypothesized diseases h_i from the domain of diseases H and multiple symptoms e_n from the set E of possible symptoms. When we consider each disease from m hypotheses H and each symptom from n pieces of evidence from E singly, we have $m \times n$ measures to collect and integrate. (Actually $m \times n$ posterior probabilities plus $m + n$ prior probabilities.)

Unfortunately, our analysis is about to get much more complex. To this point, we considered each symptom e_i individually. In actual situations, single symptoms are rarely the case. When a doctor is considering a patient, for instance, there are often many combinations of symptoms she must consider. We require a form of Bayes' theorem to consider any single hypothesis h_i in the context of the union of multiple symptoms e_i.

$$p(h_i|(e_1 \cup e_2 \cup \ldots \cup e_n)) = (p(h_i)\, p((e_1 \cup e_2 \cup \ldots \cup e_n)|h_i)) \,/\, p(e_1 \cup e_2 \cup \ldots \cup e_n)$$

The term $p(e_1 \cup e_2 \cup \ldots \cup e_n)$ does not affect the conditional probability on the left hand side of the above equation. It is simply a normalizing constant and we can eliminate it from our calculations. With one disease and a single symptom we needed only $m \times n$ measurements. Now, for every pair of symptoms e_i and e_j and a particular disease hypothesis h_i, we need know $p(ei \cup ej \mid hi)$. There will be n such observations, when there are n symptoms in E. Now, if we want to use Bayes, there will be about $m \times n$ (conditional probabilities) $+\ n$ (symptom probabilities) $+\ m$ (disease probabilities) or about $m \times n + n + m$ pieces of information to collect. This number can become very large in realistic medical systems.

In many diagnostic situations, we must also deal with negative information, e.g., when a patient does not have a symptom such as bad blood pressure. We require both:

$$\text{not}(\ p(e_i)\ ) = 1 - p(e_i) \text{ and } \text{not}(\ p(h_i \mid e_i)\ ) = 1 - p(h_i \mid e_i).$$

We also note that $p(e_i \mid h_i)$ and $p(h_i \mid e_i)$ are not the same and will almost always have different values. These relationships, and the avoidance of circular reasoning, is important for the design of Bayesian belief networks.

A final problem, noted briefly earlier, is the need to rebuild probability measures when new relationships between hypotheses and evidence sets are discovered. In many active research areas such as medicine, new discoveries happen continuously. Bayesian reasoning requires complete and up-to-date probabilities, including joint probabilities, if its conclusions are to be correct. Such extensive data collection and verification may be expensive.

Where these assumptions are met, however, Bayesian approaches offer the benefit of a mathematically well-founded handling of uncertainty. Most expert system domains do not meet these requirements and must rely on heuristic approaches, such as Stanford certainty theory, presented next. Furthermore, due to complexity issues, we know that even fairly powerful computers cannot use full Bayesian techniques for successful real-time problem solving.

2.2 An Alternative to Bayes: The Stanford Certainty Factor Algebra

Stanford certainty theory, a calculus for subjective probability measures, is based on a number of observations. The first is that in traditional probability theory, the sum of the probability for a relationship and probability against the same relationship must add to one. However, it is often the case that a human expert might have "confidence" 0.7 (say) that some relationship is true and have no feeling at all of it being not true. A further assumption that underpins certainty theory is that the knowledge content of the rules is much more important than the algebra for computing the confidences. Confidence measures correspond to the informal evaluations that human experts attach to their conclusions, such as "it is probably true", "it is almost certainly true", or "it is highly unlikely". Furthermore, they make absolutely no claims about a statistics based relationship between the condition of a rule and its corresponding action!

The Stanford certainty theory makes some simple assumptions for creating confidence measures and has some equally simple rules for combining these confidences as the program moves toward its conclusion. The first assumption is to split "confidence for" from "confidence against" a relationship:

Call $MB(H | E)$ the measure of belief of a hypothesis H given evidence E.
Call $MD(H | E)$ the measure of disbelief of a hypothesis H given evidence E.

Now either:

$1 > MB(H | E) > 0$ while $MD(H | E) = 0$, or
$1 > MD(H | E) > 0$ while $MB(H | E) = 0$.

These two measures constrain each other in that a given piece of evidence is either for or against a particular hypothesis, an important difference between certainty theory and probability theory. Once the link between measures of belief and disbelief has been established, they may be tied together again, by:

$$CF(H | E) = MB(H | E) - MD(H | E).$$

As the certainty factor (CF) approaches 1, the evidence is stronger for a hypothesis; as CF approaches -1, the confidence against the hypothesis gets stronger; and a CF around 0 indicates that either little evidence exists for or against the hypothesis or that the evidence for and against the hypothesis is balanced.

When experts put together a rule base, they must agree on a CF to go with each rule. This CF reflects their confidence in the rule's reliability. Certainty measures may be adjusted to tune the system's performance, although slight variations in the confidence measure tend to have little effect on the overall running of the system. This second role of certainty measures confirms the belief that "the knowledge gives the power," that is, the integrity of the knowledge itself best supports the production of correct diagnoses.

Because the architecture for expert systems is a production system, the premises for each rule are formed of ands and ors of a number of facts. When a production rule is used, the certainty factors associated with each condition of the premise are combined to produce a certainty measure for the overall premise as follows. For P1 and P2 premises of the rule:

CF(P1 and P2) = MIN(CF(P1), CF(P2)), and
CF(P1 or P2) = MAX(CF(P1), CF(P2)).

The combined CF of the premises, using the above rules, is then multiplied by the CF of the rule itself to get the CF for the conclusions of the rule. For example, consider the rule in a knowledge base:

(P1 and P2) or P3 → R1 (.7) and R2 (.3)

where P1, P2, and P3 are premises and R1 and R2 are the conclusions of the rule, having CFs 0.7 and 0.3, respectively. These numbers are attached to the rule when it is designed and represent the expert's confidence in the conclusion if all the premises are known with complete certainty. If the running program has produced P1, P2, and P3 with CFs of 0.6, 0.4, and 0.2, respectively, then R1 and R2 may be added to the collected case-specific results with CFs 0.28 and 0.12, respectively. Here are the calculations for this example:

CF(P1(0.6) and P2(0.4)) = MIN(0.6,0.4) = 0.4.
CF((0.4) or P3(0.2)) = MAX(0.4,0.2) = 0.4.

The CF for R1 is 0.7 in the rule, so R1 is added to the set of case-specific knowledge with the associated CF of (0.7) x (0.4) = 0.28. The CF for R2 is 0.3 in the rule, so R2 is added to the set of case-specific knowledge with the associated CF of (0.3) x (0.4) = 0.12.

One further measure is required: how to combine multiple CFs when two or more rules support the same result R. This rule reflects the certainty theory analog of the probability theory procedure of multiplying probability measures to combine independent evidence. By using this rule repeatedly one can combine the results of any number of rules that are used for determining a result R. Suppose CF(R1) is the present certainty factor associated with result R and a previously unused rule produces result R (again) with CF(R2); then the new CF of R is calculated by:

CF(R1) + CF(R2) - (CF(R1) x CF(R2)) when CF(R1) and CF(R2) are positive,
CF(R1) + CF(R2) + (CF(R1) x CF(R2)) when CF(R1) and CF(R2) are negative,
and (CF(R1) + CF(R2)) / (1 – MIN (| CF(R1) |) , | CF(R2) |) otherwise, where | X | is the absolute value of X and MIN (a , b) means the minimum value of a or b.

Besides being easy to compute, these combination equations have other desirable properties. First, the CFs that result from applying this rule are always between 1 and -1. Second, the result of combining contradictory CFs is that they cancel each other, as is desired. Finally, the combined CF measure is a monotonically increasing (decreasing) function in the manner one would expect for combining evidence.

Finally, the confidence measures of the Stanford certainty factor tradition are a human (subjective) estimate of symptom/cause probability measures. As noted in Section 2.1, in the Bayesian tradition if A, B, and C all influence D, we need to isolate and appropriately combine all the prior and posterior probabilities, including P(D), P(D|A), P(D|B), P(D|C), P(A|D), when we want to reason about D. The Stanford certainty factor tradition allows the knowledge engineer to wrap all these relationships together into one confidence factor, CF, attached to the rule; that is, if A and B and C then D (CF). It is felt that this simple algebra better reflects how human experts combine and propagate multiple sets of beliefs.

Certainty theory may be criticized as being excessively ad hoc. Although it is defined in a formal algebra, the meaning of the certainty measures is not as rigorously founded as is formal probability theory. However, certainty theory does not attempt to produce an algebra for "correct" reasoning. Rather it is the "lubrication" that lets the expert system combine confidences as it moves along through the problem at hand. Its measures are ad hoc in the same sense that a human expert's confidence in his or her results is approximate, heuristic, and informal. When MYCIN is run, for example, the CFs are used in the heuristic search to give a priority for goals to be attempted and a cutoff point when a goal need not be considered further. But even though the CF is used to keep the program running and collecting information, the power of the program remains invested in the quality of the rules.

Section 3: Graphical Models for Uncertain Reasoning

In most of AI the primary inference mechanism in stochastic domains is some form of Bayes' rule. As we noted in Section 2.1, however, the full use of Bayesian inference in complex domains quickly becomes intractable. In the following sections we present several inference techniques specifically designed to address this complexity; these include Bayesian belief networks (BBNs), Markov models, and hidden Markov models (HMMs).

3.1 The Bayesian Belief Network

The two preceding sections describe the origins of and software architecture for the rule-based expert system. It is not the purpose of this present chapter to critique the "expert system experiment" in the context of software applications. It is sufficient to mention that in many situations, especially where the problem domain was well understood, the expert system technology proved successful. In many of these situations, once a solution was achieved, the expert system technology simply morphed into the larger suite of successful software systems. Where problems were imprecise, ill defined, and ambiguous, however the expert system approach could prove both brittle and cumbersome. This is because the semantics (basic meaning entailed) through use of expert systems technology is rather weak if it exists at all. For example, the expert system designed to help sick people simply does not *understand* sick people. In summary, where the technology fit it worked well, in other contexts it was often unsuccessful.

Although Bayesian probability theory, as discussed in Section 2.1, offers a mathematical foundation for reasoning under uncertain conditions, the complexity encountered in applying it to realistic problem domains can be quite prohibitive. Fortunately, we can often prune this complexity by focusing search on a smaller set of more highly relevant events and evidence. One approach, Bayesian belief networks (Pearl 1988), offers a computational model for reasoning to the best explanation of a set of data in the context of the expected causal relationships of a problem domain.

Bayesian belief networks (BBNs) can dramatically reduce the number of parameters of the full Bayesian model and show how the data of a domain (or even the absence of data) can partition and focus reasoning. Furthermore, the modularity of a problem domain often allows the program designer to make many independence assumptions not allowed in a full Bayesian treatment. In most reasoning situations, it is not necessary to build a large joint probability table in which the probabilities for all possible combinations of events and evidence are listed. Rather, human experts seem to select the local phenomena that they know will interact and obtain probability or influence measures that reflect only these clusters of events. Experts assume all other events are either conditionally independent or that their correlations are so small that they may be ignored.

As an example Bayesian belief network, consider the traffic problem presented in Figure 3.1. Suppose that you are driving an automobile in rural New Mexico. Suddenly you begin to slow down with the traffic. You begin to wonder what the traffic problem might be. Because you have driven quite a bit in rural New Mexico you have a set of prior expectations for bad traffic, related mainly to highway construction and traffic accidents. Thus, our Bayesian Belief net representation of these prior expectations are reflected in Figure 3.1, where road construction is C, an accident is A, the presence of orange barrels is B, bad traffic is T, and flashing lights is L. To calculate the joint probability of all the parameters of the example required knowledge or measurements for all parameters being in particular states. Thus, the joint probability is:

$$p(C,A,B,T,L) = p(C) \times p(A|C) \times p(B|C,A) \times p(T|C,A,B) \times p(L|C,A,B,T)$$

The number of parameters in this joint probability is 31. This table is exponential in the number of variables involved. For a problem of any complexity, say with thirty or more variables, the joint distribution table would have more than a billion elements (see discussion in Section 2.1).

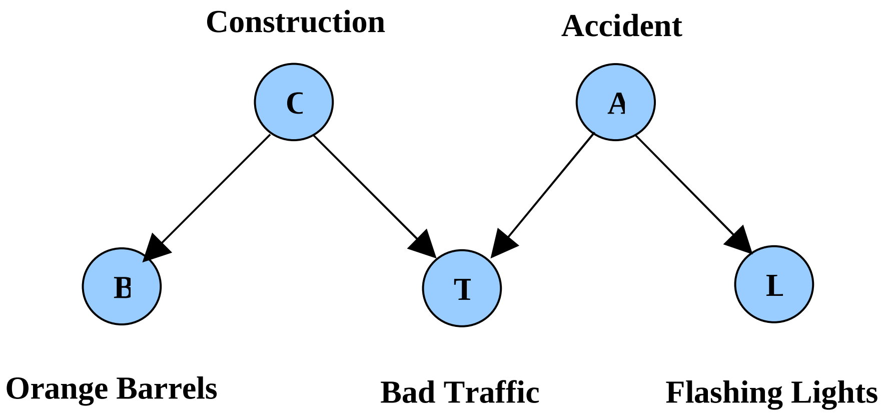

Figure 3.1 The graphical model for the traffic problem, adapted from Luger (2005).

Note, however, that if we can support the assumption that the parameters of this problem are only dependent on the probabilities of their parents, that is, we can assume that nodes are independent of all non-descendents, given knowledge of their parents, the calculation of $p(C,A,B,T,L)$ becomes:

$$p(C,A,B,T,L) = p(C) \times p(A) \times p(B|C) \times p(T|C,A) \times p(L|A)$$

To better see the simplifications we have made, consider $p(B|C,A)$ from the previous equation. We have reduced this to $p(B|C)$ in our most recent equation. This is based on the assumption that road construction is not a causal effect of there being an accident. Similarly, the presence of orange barrels is not a cause of bad traffic, but construction and accident are, giving as a result $p(T|C,A)$ rather than $p(T|C,A,B)$. Finally, $p(L|C,A,B,T)$ is reduced to $p(L|A)$! The probability distribution for $p(C,A,B,T,L)$ now has only 20 (rather than 32) parameters. And if we move to a more realistic problem, with 30 variables say, and if each state has at most two parents, there will be at most 240 elements in the distribution. If each state has three parents, the maximum is 490 elements in the distribution: considerably less than the exponentially large number required for the full Bayesian approach!

We need to justify this dependence of a node in a belief network on its parents alone. Links between the nodes of a belief network represent the conditioned probabilities for causal influence. Implicit in expert reasoning using causal inference is the assumption that these influences are directed, that is, the presence of some event somehow causes other events in the network. Further, causal influence reasoning is not circular in that some effect cannot circle back to cause itself. For these reasons, Bayesian belief networks will have a natural representation as a directed acyclic graph or DAG (Luger 2005, Section 3.1), where coherent patterns of reasoning are reflected as paths through cause/effect relationships. Bayesian belief networks are one instance of what are often called graphical models.

In the case of our traffic example we have an even stronger situation that allows us to calculate very simply the probability distribution at every node. The distributions of nodes having no parents are directly looked up. The values of child nodes are computed using only the probability distributions of each child's parents by doing the appropriate computations on the child's conditional probability table and the parent's distributions. This is possible because we don't have to worry about relationship between the parents of any node (since the network is given as a directed and acyclic graph). This produces a natural abductive separation where accident has no correlation at all with the presence of orange barrels, as is seen in Figure 1.7. We summarize our discussion of BBNs and the traffic example with the following definition.

A *Bayesian belief network* may be defined:

A graphical model is called a *Bayesian belief network* (BBN) if its graph, annotated with conditional probabilities, is directed and acyclic. Furthermore, BBNs assume nodes are independent of all their non-descendents, given knowledge of their parents.

A *dynamic Bayesian network* (DBN) is a sequence of identical Bayesian networks whose nodes are linked in the (directed) dimension of time. We consider the general DBM briefly in Section 4.1; for further details see Friedman (1998) or Ghahramani and Jordan (1997).

3.2 Inference with a Bayesian Belief Network

We refer again and extend the Bayesian belief net example of Section 3.1. Again, suppose you are driving the interstate highway system and realize you are gradually slowing down because of increased traffic congestion. You begin to search for possible explanations of the slowdown. Could it be road construction? Has there been an accident? Perhaps there are other possible explanations. After a few minutes you come across orange barrels at the side of the road that begin to cut off the outside lane of traffic. At this point you determine that the best explanation of the traffic congestion is most likely road construction. At the same time the alternative hypothesis of an accident is explained away. Similarly if, you would have seen flashing lights in the distance ahead, such as those from a police vehicle or an ambulance, the best explanation for traffic slowdown, given this new evidence, would be a traffic accident and road construction would have been explained away. When a hypothesis is explained away that does not mean that it is no longer possible. Rather, in the context of new evidence, it is simply less likely.

Figure 3.1 presented a Bayesian network account of what we have just seen. Road construction is correlated with orange barrels and bad traffic. Similarly, accident correlates with flashing lights and bad traffic. We next examine Figure 3.1 and build a joint probability distribution for the road construction and bad traffic relationship. We simplify both of these variables to be either true (t) or false (f) and represent the probability distribution in Figure 3.2. Note that if construction is f there is not likely to be bad traffic and if it is t then bad traffic is likely. Note also that the probability of road construction on the interstate, C = true, is 0.8 and the probability of having bad traffic, T = true, is 0.78.

We next consider the change in the probability of road construction given the fact that we have experienced bad traffic, or p(C|T) or p(C = t | T = t). The following equation reflects the new probability, where the numerator is the outcome we have (C = t and T = t) and the denominator reflects all possible outcomes for this situation (the sum of C = t and T = t plus C = f and T = t):

p(C|T) = p(C = t , T = t) / (p(C = t , T = t) + p(C = f , T = t)) = 0.72 / (0.72 + 0.06) = 0.923

C is true = 0.80

C	T	P
t	t	0.72
t	f	0.08
f	t	0.06
f	f	0.14

T is true = 0.78

Figure 3.2 The joint probability distribution for the traffic and the construction variables of 3.1.

So now, with the normal probability of road construction being 0.8, given that there actually is bad traffic, the probability for road construction goes up to 0.923!

Consider the expanded probability distribution table of Figure 3.3. This is an extension of Figure 3.2 as the probabilities of Figure 3.2 remain the same for all values of C and T in Figure 3.3. We now want to determine the new probability for construction C given that we are experiencing bad traffic T and see yellow barrels, B.

p(C|T,B) = p(C = t , T = t, B = t) / (p(C = t , T = t, B = t) + p(C = f , T = t, B = t)) = 0.576 / (0.576 + 0.012) = 0.98

C	T	B	P
t	t	t	0.576
t	t	f	0.144
t	f	t	0.064
t	f	f	0.016
f	t	t	0.012
f	t	f	0.048
f	f	t	0.028
f	f	f	0.112

Figure 3.3 The probability measure for Construction, Traffic and Barrels. Note the result on Construction from marginalizing across the situations where T = t and B = t.

This new calculation of p(C|T,B) shows that the traffic slowdown plus the presence of yellow traffic control barrels makes the new probability of construction even higher! This is the insight supporting the Bayesian belief net technology. The priors of the situation represent our ongoing expectations of the "state of the world". When new information appears, such as the yellow traffic control barrels, then our current expectations for the "state of the world" change. Bayesian inference captures these changing expectations quite naturally.

The next example, adapted from Pearl (1988), shows a more complex Bayesian network.

In Figure 3.4, the season of the year determines the probability of rain as well as the probability of water from a sprinkler system. The wet sidewalk will be correlated with rain or water from the sprinkler. Finally, the sidewalk will be slick depending on whether or not it is a wet sidewalk. In the figure we have expressed the probability relationship that each of these parameters has with its parents. Note also that, as compared with the traffic example, the slippery sidewalk example has an undirected cycle (a cycle in the underlying undirected graph).

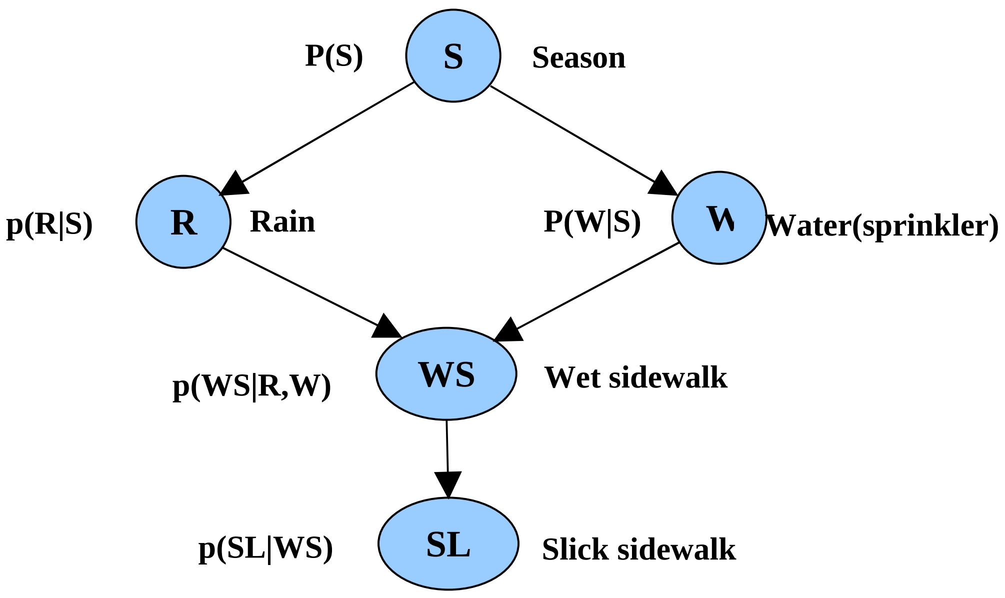

Figure 3.4 An example of a Bayesian probabilistic network, where the probability dependencies are located next to each node. This example is from Pearl (1988).

We can now ask the question, how can the probability of wet sidewalk, p(WS), be described? It can't be done as previously, where p(W) is equal to p(W|S) * p(S) or p(R) is equal to p(R|S) * p(S). The two causes of WS are not independent given S; for example, if S is summer, then p(W) and p(R) could both go up. Thus the complete conditional probabilities of the two variables, along with their further relation to S, must be calculated. In this situation we can do it, but as we will see, this calculation is exponential in the number of possible causes of WS. The calculation is represented in Figure 3.5, where we calculate one entry in that table, p(WS) where R and W are both true. To make life simpler we assume the season S is either hot or cold.

```
p(WS)  = p( R = t, W = t) for all conditions of S, season
             = p( S = hot) * p( R = t | S = hot) * p( W = t | S = hot) +
               p( S = cold) * p( R = t | S = cold) * p( W = t | S = cold)
```

In a similar fashion the remainder of Figure 3.5 can be completed. This makes up the joint probability for rain and water from sprinkler. This larger "macro element" represents p(WS) = p(WS | R,W) * p(R,W). We have gotten away with a rather reasonable calculation, the problem is that this calculation is exponential in the number of parents of the state.

R	W	WS
t	t	p(WS) {
t	f	
f	t	
f	f	

S = hot
S = cold

Figure 3.5 The probability distribution for p(WS), a function of p(W) and p(R), given the effect of S. We calculate the effect when R = t and W = t.

We call this macro element the *combined variable*, or *clique*, for the calculation of p(WS). We employ this concept of a clique in order to replace the constraint propagation of the DAG of Figure 3.4 with an acyclic clique tree, as seen in Figure 3.6. The rectangular boxes of Figure 3.6a reflect the variables that the cliques above and below it share. The table that passes the relevant parameters through to the next clique is exponential in the number of these parameters. It should also be noted that a linking variable along with all its parents must be present in the clique. Thus, in setting up a belief network or other graphical model (the knowledge engineering process), we ought to be careful how many variables are parents of any state. The cliques will also overlap, as seen in Figure 3.6b, to pass information through the full tree of cliques, called the junction tree. We next present a algorithm developed by Lauritzen and Spiegelhalter (1988) that creates a junction tree from any Bayesian belief network.

1. For all nodes in the belief network make all directed links undirected.

2. For any node draw links between all its parents (the dashed line between R and W in Figure 3.6b).

3. Look for any cycle in the resulting graph of length more than three and add further links that reduce that cycle to three. This process is called triangulation and is not necessary in the example of Figure 3.6b.

4. Create the junction tree from the triangulated structure of step 3 by finding the maximal cliques (cliques that are complete subgraphs and not subgraphs of a larger clique). The variables in these cliques are put into junctions and the junction tree is created by connecting any two junctions that share at least one variable, as in Figure 3.6a.

The triangulation process described in step 3 above is critical, as we want the resulting junction tree to have minimal computational cost when propagating information. Unfortunately, this decision of designing optimal cost junction trees is NP hard. Often, fortunately, a simple greedy algorithm can be sufficient for producing useful results. Note that the sizes of the tables required to convey information across the junction tree of Figure 3.6 are 2*2*2, 2*2*2, and 2*2.

A final comment: Bayesian belief networks seem to reflect how humans reason in complex domains where some factors are known and related a priori to others. As reasoning proceeds by progressive instantiation of information, search is further restricted, and as a result, more efficient. This search efficiency stands in strong contrast to the approach supported by using a full joint distribution, where more information requires an exponentially larger need for statistical relations and a resulting broader search.

Many algorithms exist for building belief networks and propagating arguments as new evidence is acquired. We recommend especially Pearl's (1988) message passing approach and the *clique tree triangulation* method proposed by Lauritzen and Spiegelhalter (1988). Druzdel and Henrion (1993) have also proposed algorithms for propagating influence in a network. Dechter (1996) presents the *bucket elimination* algorithm as a unifying framework for probabilistic inference.

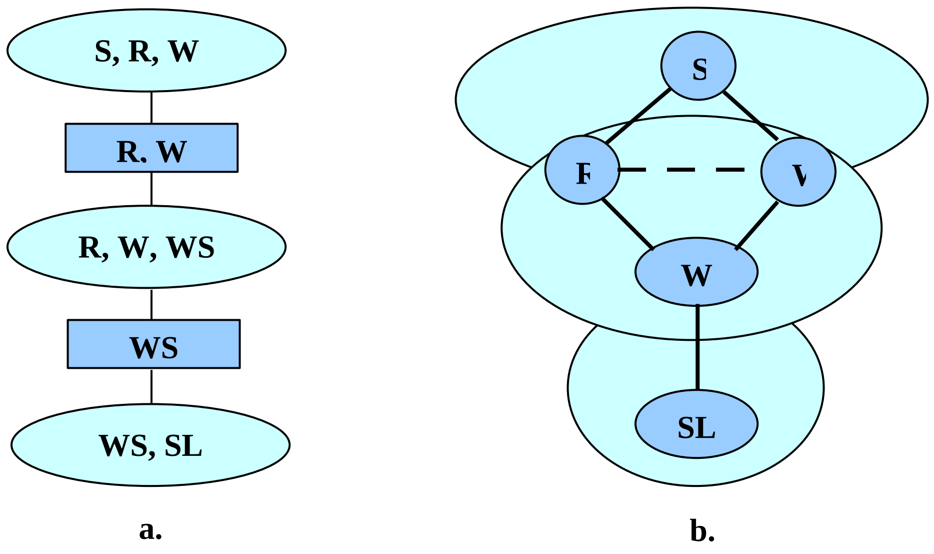

Figure 3.6 A junction tree (a) for the Bayesian probabilistic network of (b). Note that we started to construct the transition table for the rectangle R, W, adapted from Luger (2005).

Section 4: Expert Systems and Graphical Models: The Continuing Story

This chapter has summarized the evolution of expert systems and graphical models over the last several decades. Our goal has been to give a top-down description of this technology and to describe, without overwhelming detail, the key features of both its evolution and use. We conclude our presentation by giving short introductions to several other technologies that have come to extend and sometimes replace the traditional expert system. These include Markov models, hidden Markov models, and dynamic Bayesian networks.

4.1 Diagnostic and Prognostic Reasoning with Graphical Models

Figure 4.1 presents a Markov state machine (sometimes called a Markov chain) with four distinct states. This general class of systems may be described at any time as being in one of a set of n distinct states, s_1, s_2, s_3, ... , s_n. The system undergoes changes of state, with the possibility of it remaining in the same state, at regular discrete time intervals. We describe the ordered set of times t that are associated with the discrete intervals as t_1, t_2, t_3, ... , t_n. The system changes state according to the distribution of probabilities associated with each state. We denote the actual state of the machine at time t as s_t.

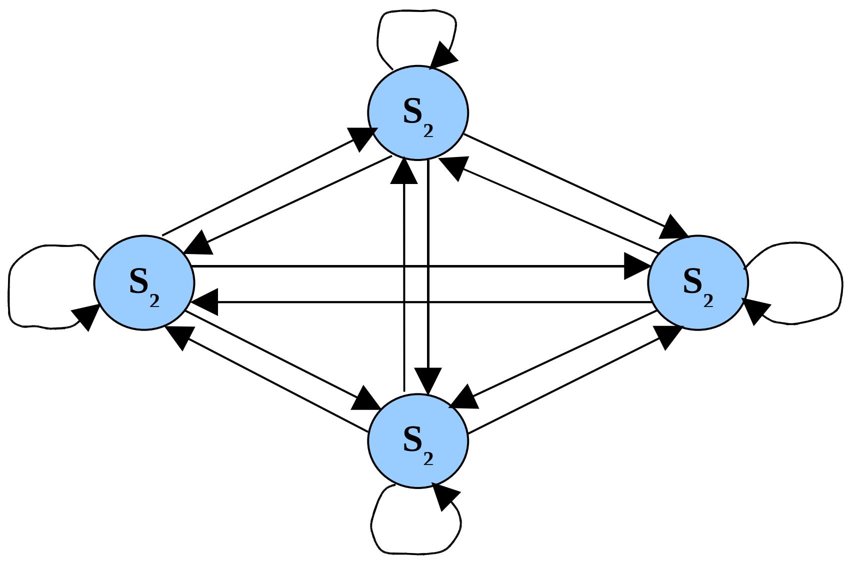

Figure 4.1 A Markov state machine or Markov chain with four states, S_1, S_2, S_3 and S

A full probabilistic description of this system requires, in the general case, the specification of the present state s_t, in terms of all its predecessor states. Thus, the probability of the system being in any particular state s_t is:

$$p(s_t) = p(s_t \mid s_{t-1}, s_{t-2}, s_{t-3}, \ldots)$$

where the s_{t-1} are the predecessor states of s_t. In a *first-order Markov chain*, the probability of the present state is a function only of its direct predecessor state:

$$p(s_t) = p(s_t \mid s_{t-1})$$

where s_{t-1} is the predecessor of s_t. We next assume that the right side of this equation is time invariant, that is, we hypothesize that across all time periods of the system, the transitions between specific states retain the same probabilistic relationships.

Based on these assumptions, we now can create a set of state transition probabilities a_{ij} between any two states s_i and s_j as follows:

$$a_{ij} = p(s_t = s_i \mid s_{t-1} = s_j), \quad 1 \geq i, j \geq N$$

Note that i can equal j, in which case the system remains in the same state. The traditional constraints remain on these probability distributions; for each state s_i:

$$a_{ij} \geq 0, \text{ and for all } j, \sum_{i=1}^{N} a_{ij} = 1$$

The system we have just described is called a first-order observable Markov model since the output of the system is the set of states at each discrete time interval, and each state of the system corresponds to a physical (observable) event. We make the observable Markov model more formal with a definition (as follows) and then give an example.

An *(observable) Markov model* may be defined:

A graphical model is called an *(observable) Markov model* if its graph is directed and the probability of arriving at any state s_t from the set of states S at a discrete time t is a function of the probability distributions of its being in previous states of S at previous times. Each state s_t of S corresponds to a physically observable situation.

An observable Markov model is *first-order* if the probability of it being in the present state s_t at any time t is a function only of its being in the previous state s_{t-1} at the time $t - 1$, where s_t and s_{t-1} belong to the set of observable states S.

As an example of an observable first-order Markov model, consider the weather at noon, say, for a particular location. We assume this location has four different discrete states for the variable weather: s_1 = sun, s_2 = cloudy, s_3 = fog, s_4 = precipitation. We assume that the time intervals for the Markov model will be noon each consecutive day. We also assume the transition probabilitiess between the states of weather remain constant across time (not true for most locations!), and that the observable, weather, can remain in the same state over multiple days. This situation is represented by Figure 4.1, and is supported by the matrix of state transitions a_{ij}:

$$a_{ij} = \begin{array}{ccccc} & s_1 & s_2 & s_3 & s_4 \\ s_1 & 0.4 & 0.3 & 0.2 & 0.1 \\ s_2 & 0.2 & 0.3 & 0.2 & 0.3 \\ s_3 & 0.1 & 0.3 & 0.3 & 0.3 \\ s_4 & 0.2 & 0.3 & 0.3 & 0.2 \end{array}$$

In this a_{ij} transition matrix the first row represents the transition probabilities from s_1 to each of the states, including staying in the same state; the second row is the transition probabilities from s_2 to each of the states, and so on. Note that the properties required for the transition probabilities to be probability distributions from each state are met (they sum to 1.0).

We now can ask questions of our model. Suppose that today, s_1, is sun; what is the probability of the next five days remaining sun? Or again, what is the probability of the next five days being sun, sun, cloudy, cloudy, precipitation? We solve this second problem. We wish to determine the probability of observing, given our model, the set of states, where the first day, s_1, is today's observed sunshine:

$$O = s_1, s_1, s_1, s_2, s_2, s_4$$

The probability of this sequence of observed states, given the first-order Markov model, M, is:

$$\begin{aligned} p(O \mid M) &= p(s_1, s_1, s_1, s_2, s_2, s_4 \mid M) \\ &= p(s_1) \times p(s_1 \mid s_1) \times p(s_1 \mid s_1) \times p(s_2 \mid s_1) \times p(s_2 \mid s_2) \times p(s_4 \mid s_2) \\ &= 1 \times a_{11} \times a_{11} \times a_{12} \times a_{22} \times a_{24} \\ &= 1 \times (.4) \times (.4) \times (.3) \times (.3) \times (.3) \\ &= .00432 \end{aligned}$$

This equation follows from the assumptions of the first-order Markov model. Thus the state of weather for each day is a function (only) of the weather the day before and we observed the fact that today is sunshine.

We can extend this example to determine, given that we know today's weather, the probability that the weather will be the same for exactly the next t days, i.e., that the weather remains the same until the $t + 1$ day at which time it is different. For any weather state s_i, and Markov model M, we have the observation O:

$O = \{ s_i, s_i, s_i, \ldots , s_i, s_j\}$, where there are exactly $(t + 1)$ s_i, and where s_i does not equal s_j, then:

$$p(O \mid M) = 1 \times a_{ii}^{t} \times (1 - a_{ii})$$

where a_{ii} is the transition probability of taking state s_i to itself. This value is called the *discrete probability density function* for the duration of t time periods in state s_i of model M. This duration density function is indicative of the state duration in a Markov model. Based on this value we can calculate, within model M, the expected number of observations of, or duration d_i within any state s_i, given that the first observation is in that state:

$$d_i = \sum_{d=1}^{n} d \times a_{ii}^{(t-1)} \times (1 - a_{ii}) \qquad \text{where } n \text{ approaches infinity, or:}$$

$$= \frac{1}{1 - a_{ii}}$$

For example, the expected number of consecutive precipitation days, given this model M, is 1/(1 - .3) or 1.43. Similarly, the number of consecutive sunny days one might expect is 1.67.

We next consider Markov models whose states are not observable events, that is, they are themselves probabilistic functions of the state. In the Markov models we have seen to this point, each state corresponded to a discrete physical - or observable - event, such as the value of weather at a certain time of day. This class of models is really fairly limited and we now generalize it to a wider class of problems. In this section we extend Markov models to the situations where the observations are themselves probabilistic functions of a current hidden state. This resulting model, called a hidden Markov model (HMM), is a doubly embedded stochastic process.

The HMM is an observable stochastic process masking a further non-observable, or hidden, stochastic process. An example of a HMM would be to determine a phoneme (an atomic unit of voiced speech) through the interpretation of noisy acoustic signals. The *phone patterns* themselves, that is, which phonemes are more likely to follow others in the particular words of a language, make up the hidden level of the Markov model. The *observations*, i.e., the noisy acoustic signals, are a stochastic function of these phonemes. The phoneme level of the model cannot be "seen" except though the top level stream of acoustic signals.

A *hidden Markov model* may be defined as follows:

A graphical model is called a *hidden Markov model* (HMM) if it is a Markov model whose states are not directly observable but are "hidden" by a further stochastic system interpreting their output. More formally, given a set of states $S = s_1, s_2, \ldots, s_n$, and given a set of state transition probabilities $A = a_{11}, a_{12}, \ldots, a_{1n}, a_{21}, a_{22}, \ldots, \ldots, a_{nn}$, there is a set of observation likelihoods, $O = p_i(o_t)$, each expressing the probability of an observation o_t (at time t) being from a state i.

For example, consider the problem of N urns, each urn containing a collection of M differently colored balls. The physical process of obtaining observations is, according to some random process, to pick one of the N urns. Once an urn is selected a ball is removed and its color is recorded in the observable output stream. The ball is then replaced and the random process associated with the current urn selects the next (which might be the same) urn to continue the process. This process generates an observable sequence consisting of a number of colors (of the balls).

It is obvious that the simplest HMM corresponding to this ball selection process is the model in which each state corresponds to a specific urn, the values of the transition matrix for that state produce the next state choice, and in which the ball color probability is defined for each state.

A second example of a HMM is taken from the authors own use of hidden models to detect failures in helicopter rotor systems (Chakrabarti 2005). Suppose that we have three hidden states intended to describe the current condition of the transmission of a helicopter. The three states of the transmission are *safe*, *questionable*, and *faulty*. These states are hidden in the sense that they are not directly observable by the human diagnostician – except, of course, when the helicopter is a pile of rubble on the ground! We would like to make a probabilistic estimate of the state of the helicopter transmission without being able to directly observe it. We do have lots of observable data for this system, however including the current temperatures of certain components and the vibration measurements for others. See an example set of raw time series data parameters in Figure 4.2. We first use a number of data processing tools (from Mathlab), especially the Fast Fourier Transform to change the data to a time-frequency domain. Correlation plots across time periods were then utilized to note radical changes in the sampled data. These changes were used to determine the "hidden" states of the model already described and presented in Figure 4.3.

Finally, this type model, monitoring changes of a system across time periods, can also be seen as an instance of a *dynamic Bayesian network*. The dynamic Bayesian network approach to diagnosis is represented by a sequence of Bayesian networks each of whose states are mapped to themselves across time periods. That is, the changes of a system are described through the changes of the particular states of the Bayesian network across time intervals. Thus, the diagnosis of the state of the helicopter rotor system just described, Figure 4.3, may be seen as the changes of the states of the network across time, for example going from *safe* to *questionable* to *faulted*. For further details, see Chakrabarti (2005).

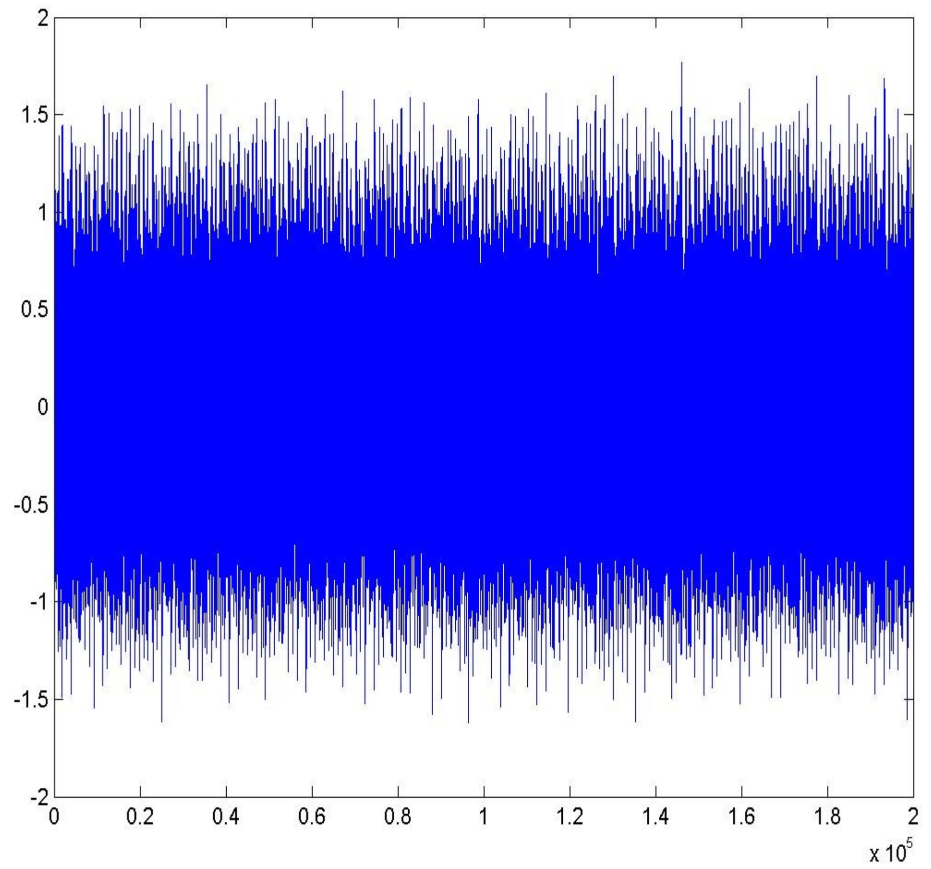

Figure 4.2 Raw time series data obtained from the sensors monitoring the helicopter rotor system.

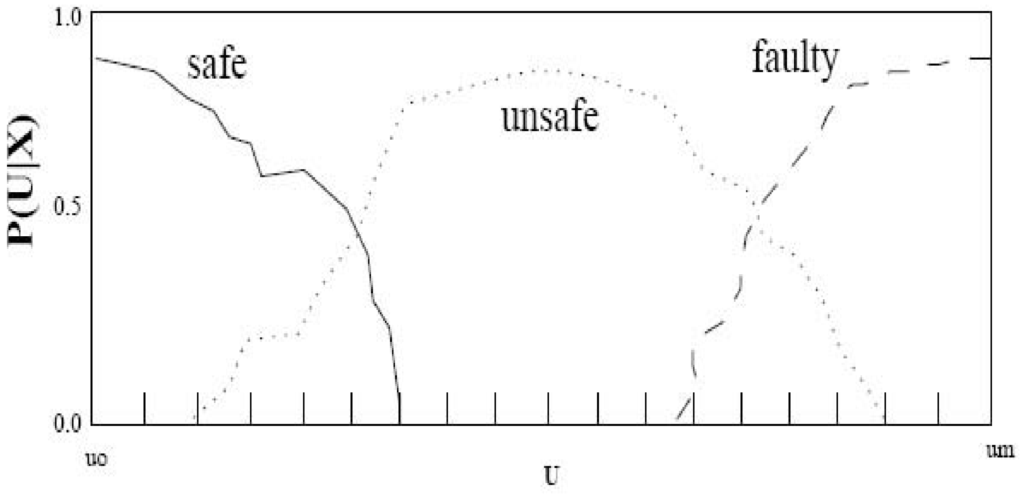

Figure 4.3 A probability distribution along the three states of the helicopter rotor system: safe, unsafe and faulty.

4.2 Graphical Models: Some Thoughts about the Future

Finally, we mention some current research that is expanding the expressive flexibility of Bayesian networks. Although there are many research institutions involved in this work world wide, our primary reference will be to our own work, especially Luger (2005) and Pless et al. (2006).

First, it should be noted that the nodes of the traditional Bayesian belief networks are *propositional* in nature. Being propositional they can only represent the likelihood that two concrete individuals or situations can be related to each other according to some distribution. We saw this with the road/traffic example of Section 3. It is often desired to extend the Bayesian representation to *prepositional* or variable-based relationships. For example we might wish to represent the fact that "all automobile transmissions" have specific failures (with a distribution, of course). This requires a predicate calculus based representational scheme. A number of research groups, including Pless et al (2006) and Kohler and Pfeffer (1998) are currently building these (so called) *first-order* representations for graphical models.

A second extension to the traditional Bayesian graphical model is to make them fully recursive and "Turing complete", that is, powerful enough to compute any function that is computable. This means that their inference scheme will be able to compute a broader range of probabilistic outcomes. It also greatly broadens the types of models that may be computed – for example dynamic Bayesian networks. The authors, (Pless et al. 2006), used exactly this approach for diagnosing faults in helicopter transmission systems, as was presented in the previous Section.

Finally, many research groups are attempting to do structured probabilistic model induction. What this means is that component facts and rules that will make up possible models of a problem domain can be stated in a declarative fashion ("This fact is true with distribution P_1", "This rule relationship is true with distribution P_2", etc., etc.). Then the interpreter *will automatically construct the models most consistent* with this set of current (a priori) expectations and the current set of observable (a posteriori) facts. The general form of this model induction task still remains outside of our current technology and inference schemes. However, several research groups are addressing this challenge, including Getoor et al. (2001), Segel et al. (2001), Cussens (2001), and Pless et al. (2006).

Acknowledgements

The research group at the University of New Mexico led by Professor George F. Luger, has been investigating models for diagnostic and prognostic reasoning for the past two decades. This research includes developing various forms of expert systems, Bayesian belief networks, as well as various other forms of stochastic models. We are grateful to the National Science Foundation, the Department of Energy (through Sandia and Los Alamos National Laboratories), and the Department of Defense (through The US Air Force, and the US Navy, and Management Sciences, Inc) for their support over these years. We are also in debt to the various graduate students that have developed their research as part of our AI group at the University of New Mexico. Finally, many of the ideas and figures of this chapter were adapted from the first five editions of the book, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, whose first author is George Luger. We thank our publisher Addison-Wesley and Pearson Education for their permission to use some of this material and figures. More complete detail on the material of this chapter may be found in this AI book.

Bibliography

Bayes, T., 1763. Essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*. London: The Royal Society, 370 – 418.

Buchanan, B. H. and Shortliff, E. H., (eds.), 1984. *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading MA: Addison-Wesley.

Chakrabarti, C., 2005. First Order Stochastic Systems for Diagnosis and Prognosis. MS Thesis. Department of Computer Science, University of New Mexico.

Cussens, J., 2001. Parameter Estimation in Stochastic Logic Programs. *Machine Learning* 44:245-271.

Forgy, C. L., 1982. RETE: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem. *Artificial Intelligence*, 19(1)17-37.

Friedman, N, 1998. The Bayesian Structural EM Algorithm. In *Proceedings of the Fourteenth Conference of Uncertainty in Artificial Intelligence*, 252-26n San Francisco CA: Morgan Kaufmann.

Getoor, L., Friedman, N., Koller, D., and Pfeffer, A., 2001. Learning Probabilistic Relational Models. *Relational Data Mining*, S. Dzeroski and N. Lavorac (eds). New York: Springer.

Ghahramani, Z. and Jordan, M. I., 1997. Factorial Hidden Markov Models. *Machine Learning* 29:245-274.

Koller, D. and Pfeffer, A., 1998. Probabilistic Frame-Based Systems. In Proceedings of the Fifteenth National Conference on AI, 580-587, Cambridge MA: MIT Press.

Luger, G. F., 2005. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. London: Addison-Wesley Pearson Education.

Pearl, J., (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Los Altos CA: Morgan Kaufmann.

Pless, D. J., Chakrabarti, C., Rammohan, R., and Luger, G. F., (2006). The Design and Testing of a First-Order Stochastic Modeling Language. *Proceedings of the International Journal of AI Tools.* World Scientific Publishing Company. (to appear).

Segel E., Koller, D., and Ormoneit, D., 2001. Probabilistic Abstraction Heirarchies. *Neural Information Processing Systems (NIPS)*, Cambridge MA: MIT Press.