# 9 REASONING IN UNCERTAIN SITUATIONS

*All traditional logic habitually assumes that precise symbols are being employed. It is therefore not applicable to this terrestrial life but only to an imagined celestial existence.*

—BERTRAND RUSSELL

*It is the mark of an instructed mind to rest satisfied with that degree of precision which the nature of the subject admits, and not to seek exactness where only an approximation of the truth is possible.*

—ARISTOTLE

*So far as the laws of mathematics refer to reality they are not certain. And so far as they are certain they do not refer to reality.*

—ALBERT EINSTEIN

## 9.0 Introduction

Through most of Parts I, II, and III, our inference procedures followed the model of reasoning used in the predicate calculus: from correct premises, sound inference rules produce new, guaranteed correct conclusions. As we saw in Chapters 5 and 7 however, there are many situations that will not fit this approach; that is, we must draw useful conclusions from poorly formed and uncertain evidence using unsound inference rules.

Drawing useful conclusions from incomplete and imprecise data with unsound reasoning is not an impossible task; we do it very successfully in almost every aspect of our daily life. We deliver correct medical diagnoses and recommend treatment from ambiguous symptoms; we analyze problems with our cars or stereos; we comprehend language

statements that are often ambiguous or incomplete; we recognize friends from their voices or their gestures; and so on.

To demonstrate the problem of reasoning in ambiguous situations, consider Rule 2 from the automobile expert system presented in Section 8.2:

if
the engine does not turn over, and
the lights do not come on
then
the problem is battery or cables.

On the surface, this rule looks like a normal predicate relation to be used in sound inferencing (modus ponens). It is not, however; it is heuristic in nature. It could be possible, though very unlikely, that the battery and cables are fine but that the car simply has a bad starter motor and burned-out headlights. Failure of the engine to turn over and the lights to come on does not necessarily imply that the battery and cables are bad.

It is interesting that the *converse* of the rule is true:

if
the problem is battery or cables
then
the engine does not turn over, and
the lights do not come on.

Barring the supernatural, with a dead battery, neither the lights nor the starter will work!

Our expert system offers an example of *abductive* reasoning. Formally, abduction states that from $P \rightarrow Q$ and $Q$ it is possible to infer $P$. Abduction is an *unsound* rule of inference, meaning that the conclusion is not necessarily true for every interpretation in which the premises are true (Section 2.3).

Although abduction is unsound, it is often essential to solving problems. The "logically correct" version of the battery rule is not very useful in diagnosing car troubles since its premise, the bad battery, is our goal and its conclusions are the observable symptoms with which we must work. The rule can be used in an abductive fashion, however, as are rules in many diagnostic expert systems. Faults or diseases cause (imply) symptoms, not the other way around; but diagnosis must work from symptoms back to their causes.

In knowledge-based systems, we often attach a certainty factor to the rule to measure confidence in its conclusion. For example, the rule, $P \rightarrow Q$ (.9), expresses the belief "If you believe $P$ to be true, then you believe $Q$ will happen 90% of the time". Thus, heuristic rules can express an explicit policy for belief.

Another issue for expert system reasoning is how to draw useful results from data with missing, incomplete, or incorrect information. We may use certainty measures to reflect our belief in the quality of data, for example, asserting that the lights have full power (.2) can indicate that the headlights do come on, but are weak and barely visible. Beliefs and imperfect data can be propagated through rules to constrain conclusions.

In this chapter, we discuss several ways of managing abductive inference and uncertainty, especially as it is required for knowledge-intensive problem solving. In Section 9.1, we show how logic-based formalisms can be extended to address the abductive task, including the use of nonmonotonic systems supported by truth-maintenance algorithms. In Section 9.2, we consider several alternatives to logic, including the Stanford certainty factor algebra, "fuzzy" reasoning, and the Dempster–Shafer theory of evidence. These simple calculi are often utliized to address some of the complexity issues of using the full Bayesian approach for building expert systems.

In Section 9.3, we introduce stochastic approaches to uncertain reasoning. These techniques are founded on Bayes' theorem for reasoning about the frequency of events, based on prior information about these events. We conclude 9.3 by introducing graphical models, including Bayesian belief networks and observable and hidden Markov models. We present stochastic approaches to learning in Chapter 13.

## 9.1 Logic-Based Abductive Inference

We first present logic-based approaches to abduction. With logic, pieces of knowledge are explicitly used in reasoning, and, as we saw in Chapter 8, can be part of the explanations of derived conclusions. But traditional logic also has its limitations, especially in areas where information is missing, changing, or uncertain; in these situations traditional inference procedures may not be usable. In Section 9.1 we present several extensions to traditional logic that allow it to support abductive inference.

In Section 9.1.1, we extend logic to let it describe a world of changing information and beliefs. Traditional mathematical logic is *monotonic*: it begins with a set of axioms, assumed to be true, and infers their consequences. If we add new information to this system, it may cause the set of true statements to increase. Adding knowledge will never make the set of true statements decrease. This monotonic property leads to problems when we attempt to model reasoning based on beliefs and assumptions. In reasoning with uncertainty, humans draw conclusions based on their current set of beliefs; however, unlike mathematical axioms, these beliefs, along with their consequences, may change over time as more information becomes available.

*Nonmonotonic reasoning* addresses the problem of changing beliefs. A nonmonotonic reasoning system handles uncertainty by making the most reasonable assumptions in light of uncertain information. It then proceeds with its reasoning as if these assumptions were true. At a later time a belief may change, necessitating a re-examination of any conclusions derived from that belief. Truth maintenance algorithms, Section 9.1.2, may then be employed to keep the knowledge base consistent. Other abductive extensions to logic include "minimum models", Section 9.1.3, and the "set cover" approach, Section 9.1.4.

### 9.1.1 Logics for Nonmonotonic Reasoning

Nonmonotonicity is an important feature of human problem solving and common sense reasoning. In most planning, for example when we drive to work, we make numerous

assumptions about the roads and traffic. If we find that one of these assumptions is violated, perhaps by construction or an accident on our usual route, we change our plans and find an alternative route.

Conventional reasoning using predicate logic is based on three important assumptions. First, the predicate descriptions must be sufficient with respect to our application domain. That is, all the information necessary to solve the problem must be represented. Second, the information base must be consistent; that is, pieces of knowledge cannot contradict each other. Finally, through the use of inference rules, the known information grows monotonically. If any of these three assumptions is not satisfied, the conventional logic-based approach will not work.

Nonmonotonic systems address each of these three issues. First, reasoning systems are often faced with a lack of knowledge about a domain. There is an important issue here: suppose we have no knowledge about the predicate p; does lack of knowledge mean that *we are not sure whether p is true* or *we are sure that not p is true*? This question can be answered in a number of ways. Prolog, see Section 14.3, uses *the closed world assumption* to determine as false anything that its reasoning system cannot prove to be true. As humans, we often take the alternative approach of assuming something to be true unless it can be explicitly shown to be false.

Another approach to the lack of knowledge problem is to make explicit assumptions of truth. In human reasoning, we assume the innocence of people not directly connected to a crime. We would probably even go further and assume the innocence of those that could not benefit from the crime. The result of these assumptions is to effectively fill in missing details of knowledge and extend our reasoning to reach new conclusions based on these assumptions. We discuss the closed world assumption and its alternatives in Section 9.1.3.

Humans reason based on *how the world usually works*. Most birds fly. Parents usually love and support their children. We make inferences based on the consistency of reasoning with our assumptions about the world. In this section we discuss the addition of modal operators, such as *is consistent with* and *unless,* to perform assumption-based reasoning.

The second assumption required of traditional logic-based systems is that the knowledge supporting reasoning must be consistent. For human reasoners this would be a very limiting assumption. In diagnosing a problem we often entertain multiple possible explanations for a situation, assuming something is true until an alternative assumption proves to be more fruitful. In analysis of an airline accident, for example, a crash expert will consider a number of alternative causes, only eliminating (explaining away) some as new information is discovered. We humans use knowledge of the world *as it usually is* to try to direct reasoning through alternative scenarios. We would like logic systems to be able to entertain alternative hypotheses.

Finally, if we wish to use logic we must address the problem of how a knowledge base is updated. There are two issues here: first, how can we possibly add knowledge that is based on assumption only, and secondly, what can we do when one of our assumptions is later shown to be incorrect. To address the first issue, we can allow the addition of new knowledge based on assumptions. This new knowledge is assumed to be correct and so it may, in turn, be used to infer more new knowledge. The cost of this practice is that we must keep track of all reasoning and proofs that are based on assumptions: we must be prepared to reconsider any knowledge based on these assumptions.

Nonmonotonic reasoning, because conclusions must sometimes be reconsidered, is called *defeasible;* that is, new information may sometimes invalidate previous results. Representations and search procedures that keep track of the reasoning steps of a logic system are called *truth maintenance systems* or TMSs. In defeasible reasoning, the TMS preserves the consistency of the knowledge base, keeping track of conclusions that might later need to be questioned. We consider several approaches to truth maintenance in Section 9.1.2. We first consider operators that can make traditional logic-based reasoning systems defeasible.

In implementing nonmonotonic reasoning, we may extend our logic with the operator unless. unless supports inferences based on the belief that its argument is not true. Suppose we have the following set of predicate logic sentences:

p(X) unless q(X) → r(X)
p(Z)
r(W) → s(W)

The first rule means that we may infer r(X) if p(X) is true and we do not believe q(X) to be true. When these conditions are met, we infer r(X) and, using r(X), can then infer s(X). Subsequently, if we change our belief, or find that q(X) is true, r(X) and also s(X) must be retracted. Note that unless deals with matters of belief rather than truth. Consequently, changing the value of its argument from "either unknown or believed false" to "believed or known to be true" can cause us to retract all inferences that depend upon these beliefs. By extending our logic to reason with beliefs that may later be retracted, we introduce nonmonotonicity into the system.

The reasoning scheme just described can also be used to encode default rules (Reiter 1980). If we replace p(X) unless q(X) → r(X) with p(X) unless ab p(X) → r(X), where ab p(X) represents abnormal p(X), we state that unless we have an abnormal instance of p, such as a bird with a broken wing, we can make the inference that if X is a bird then X can fly.

A second modal operator for extending logic systems is suggested by McDermott and Doyle (1980). They augment first-order predicate logic with the modal operator M, which placed before a predicate is read as *is consistent with*. For example:

∀ X good_student(X) ∧ M study_hard(X) → graduates(X)

This clause can be read: For all X where X is a good student, and if the fact that X studies hard is consistent with everything else we know, then X will graduate. Of course, the difficult part here is defining precisely what *is consistent with everything else we know* might in fact mean.

We first note that *is consistent with everything else we know* may not be decidable. The reason is that a modal operator forms a superset of an already undecidable system, see Section 2.2.2, and will thus be undecidable. There are two ways to address undecidability. First, we can use a *negation as failure* proof to demonstrate *is consistent with*. In our example, we would attempt the proof of not(study_hard(X)) and if we couldn't prove that X doesn't study, then we assume that X does study. We often use this approach in a

Prolog-like approximation of predicate logic. Unfortunately, negation as failure may unduly restrict our domain of interpretation.

A second approach to the *is consistent with* problem is to make a heuristic-based and limited (time or memory limited) search for the truth of the predicate, in our example study_hard(X), and then, if there is no contrary evidence, assume it to be true with the understanding that we may have to later retract the graduates conclusion and all further conclusions based on it.

We can also produce potentially contradictory results using the *is consistent with* operator. Suppose a person, Peter, is a good student but also seriously enjoys parties. We might then have the following set of predicates that describe the situation:

$\forall$ X good_student(X) $\wedge$ M study_hard(X) $\rightarrow$ graduates(X)
$\forall$ Y party_person(Y) $\wedge$ M not(study_hard(Y)) $\rightarrow$ not(graduates(Y))
good_student(peter)
party_person(peter)

With this set of clauses, where we have no further information about Peter's study habits, whether he studies hard or not, we can infer both that Peter will and will not graduate!

One reasoning method that guards against such contradictory results is to keep track of the variable bindings used with the modal operator *is consistent with*. Thus, once Peter was bound to either the study_hard or the not(study_hard) predicate, the system would prevent the binding of Peter to the other predicate. Other nonmonotonic logic systems (McDermott and Doyle 1980) are even more conservative and prevent any conclusions from such potentially contradictory clause sets. We can create another anomaly:

$\forall$ Y very_smart(Y) $\wedge$ M not(study_hard(Y)) $\rightarrow$ not(study_hard(Y))
$\forall$ X not(very_smart(X)) $\wedge$ M not(study_hard(X)) $\rightarrow$ not(study_hard(X))

From these clauses we can infer a new clause:

$\forall$ Z M not(study_hard(Z)) $\rightarrow$ not(study_hard(Z))

Further developments of the semantics of the *is consistent with* operator address such anomalous reasoning. One further extension is *autoepistemic logic* (Moore 1985).

Another nonmonotonic logic system is *default logic*, created by Reiter (1980). Default logic employs a new set of inference rules of the form:

A(Z) $\wedge$ : B(Z) $\rightarrow$ C(Z)

which is read: If A(Z) is provable *and it is consistent with what we know to assume* B(Z) then we can conclude C(Z).

To this point default logic sounds much like McDermott and Doyle's nonmonotonic logic just described. An important difference between the two is the method in which reasoning is performed. In default logic, these special inference rules are used to infer sets of plausible extensions of the original axiom/theorem set. Each extension is created by using one of the default logic inferencing rules on the knowledge represented by the original axiom/theorem set. Thus, it would be natural to have a number of plausible extensions to

an original knowledge base. This can be seen with the graduates clauses:

$$\forall \text{ X good\_student(X)} \wedge \text{: study\_hard(X)} \rightarrow \text{graduates(X)}$$
$$\forall \text{ Y party(Y)} \wedge \text{: not(study\_hard(Y))} \rightarrow \text{not(graduates(Y))}$$

Each clause could be used to create a unique plausible extension based on the original set of knowledge.

Default logic then allows any theorem inferred in a plausible extension to be admitted as an axiom for further reasoning. There must be some decision-making guide to finally determine which extension is to be used for further problem solving. Default logic says nothing about how to choose among possible plausible extensions of a knowledge base. Reiter (1978), Reiter and Criscuolo (1981), and Touretzky (1986) develop these issues.

Finally, there is also a nonmonotonic reasoning situation created by inheritance search over representations where objects can inherit from more than one parent. Peter, the party-loving good student mentioned earlier, could inherit one set of properties from being a good student, i.e., that he would most likely graduate. Peter could also inherit other, and in this case partially conflicting, properties from being a party person, that is, that he would not graduate.

An important problem facing nonmonotonic reasoning systems is the task of efficiently revising a set of conclusions in the light of changing beliefs. If, for example, we use the predicate r to infer s, then removing r removes also the support for s, as well as every other conclusion that used s. Unless there is an independent set of inferences supporting s, it must be retracted. Implementing this retraction process requires, in the worst case, that we recompute all conclusions each time a belief changes. *Truth maintenance systems,* presented next, offer mechanisms for maintaining the consistency of knowledge bases.

### 9.1.2 Truth Maintenance Systems

A *truth maintenance system* (TMS) can be employed to protect the logical integrity of the conclusions of an inferencing system. As pointed out in the previous section, it is necessary to recompute support for items in a knowledge base whenever beliefs expressed by the clauses of the knowledge base are revised. Truth maintenance systems address this issue by storing justifications for each inference and then reconsidering support for conclusions in the light of new beliefs.

One way of viewing this problem is to review the backtrack algorithm first presented in Section 3.2.2. Backtracking is a systematic method for exploring all the alternatives for decision points in search-based problem solving. An important shortcoming of the backtrack algorithm, however, is the way it systematically (and blindly) backs out of dead end states of the space and looks for alternatives from its most recent choices. This approach is sometimes called *chronological backtracking*. We grant that chronological backtracking will systematically check all alternatives in the space; however, the way it proceeds is time-consuming, inefficient, and in a very large space, useless.

What we really want in logic-based search is the ability to backtrack directly to the

point in the space where the problem occurs, and to make adjustments to the solution at that state. This approach is called *dependency-directed backtracking*. Consider an example from nonmonotonic reasoning. We need to find out about p, which we cannot directly infer. There is, however, a plausible assumption q, which, if true, will support p. So we assume q and derive p. Our reasoning continues and based on p we conclude r and s. We continue on in our reasoning and conclude without the support of p, r, or s the results t and u. Finally, we prove that our earlier assumption of q is false. What are we to do?

Chronological backtracking would revisit our reasoning steps in the reverse order in which they were made. Dependency-directed backtracking would go immediately back to the source of the contradictory information, namely the first assumption of q. Then it would go forward retracting p, r, and s. We may, at this time check to see if r and s can be derived independently of p and q. Just because they were originally produced with an incorrect assumption, does not mean that they are not otherwise supported. Finally, because t and u were derived without p, r, or s, we would not need to reconsider them.

In order to use dependency-directed backtracking in a reasoning system, we must:

1. Associate with the production of each conclusion its justification. This justification indicates the derivation process for that conclusion. The justification must contain all the facts, rules, and assumptions used to produce the conclusion.

2. Provide a mechanism that, when given a contradiction along with its justification, finds the set of false assumptions within that justification that led to the contradiction.

3. Retract the false assumption(s).

4. Create a mechanism that follows up the retracted assumption(s) and retracts any conclusion that uses within its justifications the retracted false assumption.

Of course, all retracted conclusions are not necessarily false, so they must be rechecked to see if they can be justified independent of the retracted clauses. We next present two methods for building dependency directed backtracking systems.

Jon Doyle (1979) created one of the earliest truth maintenance systems, called a *justification-based truth maintenance system* or JTMS. Doyle was the first researcher to explicitly separate the truth maintenance system, a network of propositions and their justifications, from the reasoning system operating in a domain. The result of this split is that the JTMS communicates with the problem solver, perhaps an automated theorem prover, receiving information about new propositions and justifications and in turn supplying the problem solver with information about which propositions should be believed based on the current existing justifications.

There are three main operations that are performed by the JTMS. First, the JTMS inspects the network of justifications. This inspection can be triggered by queries from the problem solver such as: Should I believe in proposition p? Why should I believe proposition p? What assumptions underlie proposition p?

The second operation of JTMS is to modify the dependency network, where modifications are driven by information supplied by the problem solver. Modifications

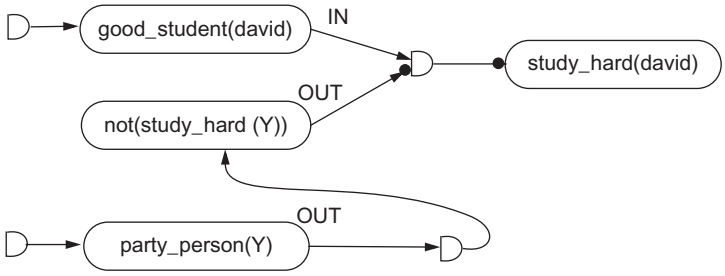


Figure 9.1 A justification network to believe that David studies hard.

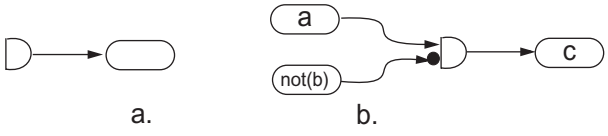


Figure 9.2 9.2(a) is a premise justification, and 9.2(b) the ANDing of two beliefs, a and not b, to support c (Goodwin 1982).

include adding new propositions, adding or removing premises, adding contradictions, and justifying the belief in a proposition. The final operation of the JTMS is to update the network. This operation is executed whenever a change is made in the dependency network. The update operation recomputes the labels of all propositions in a manner that is consistent with existing justifications.

To demonstrate the JTMS we construct a simple dependency network. Consider the modal operator M presented in Section 9.1.1, which placed before a predicate is read as *is consistent with*. For example:

$\forall$ X good_student(X) $\wedge$ M study_hard(X) $\rightarrow$ study_hard(X)
$\forall$ Y party_person(Y) $\rightarrow$ not (study_hard(Y))
good_student(david)

We now make this set of propositions into a justification network.

In a JTMS, each predicate representing a belief is associated with two other sets of beliefs. The first set, labeled IN in Figure 9.1, is the set of propositions that should be believed for the proposition to hold. The second, labeled OUT, are propositions that should not be believed for the proposition to hold. Figure 9.1 represents the justification that supports study_hard(david) derived from the predicates just listed. The notations of Figure 9.1 are adapted from Goodwin (1982) and explained in Figure 9.2. The premises of justifications are labeled as in Figure 9.2(a) and the combinations of propositions that support a conclusion are labeled as in Figure 9.2(b).

With the information of the network of Figure 9.1, the problem solver can reason that study_hard(david) is supported, because the premise good_student(david) is considered true and it is consistent with the fact that good students study hard. There is also no evidence or other indication in this example that David does not study hard.

Suppose we add the premise party_person(david). This addition enables the derivation not(study_hard(david)), and the belief study_hard(david) is no longer supported. The justifications of this situation is in Figure 9.3; note the relabeling of IN and OUT.

As Figures 9.1 and 9.3 demonstrate, the JTMS does not directly represent the predicate relationships as expressed in the original set of propositions. Rather the JTMS is a simple network that only considers the relations between atomic propositions and their negation and organizes these into support relationships for beliefs. The full set of predicate connectives and inferencing schemes ($\forall X$, $\wedge$, $\vee$, $\rightarrow$, etc.) are used within the problem solver itself. The systems of McAllester (1978) and Martins and Shapiro (1988) merged the TMS and the problem solver into a single representation.

A JTMS is only concerned with the dependencies among beliefs and has no concern with the contents of these beliefs. Therefore, we can replace the beliefs by identifiers, often of the form $n_1$, $n_2$, ..., which are associated with objects in the network called nodes. Then the algebra of INs and OUTs that we saw implemented in the study_hard example allows the JTMS to reason about the support for beliefs.

To summarize, a JTMS works with sets of nodes and justifications. Nodes stand for beliefs, and justifications support belief in nodes. Associated with nodes are the labels IN and OUT, which indicate the belief status of the associated node. We can reason about the support for any node by relating it to the INs and OUTs of the other nodes that make up its justification(s). The primary operations of the JTMS algebra is to accomplish the inspection, modification, and updating of operators noted above. Finally, since justification checking is enforced by backing over the links of the justification network itself, we have an example of dependency-based backtracking. For further information on this approach to JTMS see Doyle (1983) or Reinfrank (1989).

A second type truth maintenance system is the *assumption-based truth maintenance system* (ATMS). The term *assumption-based* was first introduced by deKleer (1984), although similar ideas may be found in Martins and Shapiro (1983). In these systems, the labels for nodes in the network are no longer IN and OUT but rather the sets of premises (assumptions) underlying their derivation. deKleer also makes a distinction between premise nodes that hold universally and nodes that can be assumptions made by the problem solver and that may later be retracted.

An advantage of ATMS over JTMS stems from the additional flexibility the ATMS provides in dealing with multiple possible states of belief. By labeling beliefs with the sets of premises under which they hold, there is no longer a single state of belief (in JTMS all the nodes labeled IN), but rather a number of possible states, the set of all subsets of the supporting premises. The creation of different belief sets, or possible worlds, allows a comparison of results from different choices of premises, the existence of different solutions for the problem, and the detection of and recovery from contradictions. The disadvantages of ATMS include the inability to represent premise sets that are themselves nonmonotonic and the control over the problem solver. However, see Dressler (1988) and Forbus and deKleer (1988) for alternatives.

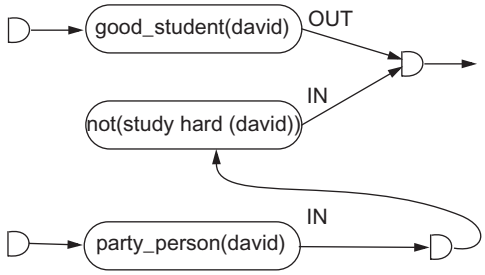


Figure 9.3 The new labeling of Figure 9.1 associated with the new premise party_person(david).

The communication between the ATMS and the problem solver is similar to that between JTMS and its problem solver with operators for *inspection*, *modification*, and *updating*. The only difference is that with the ATMS there is no longer a single state of belief but rather subsets of potential supporting premises. The goal of computation within the ATMS is to find minimal sets of premises sufficient for the support of each node. This computation is done by propagating and combining labels, beginning with the labels for the premises.

We next present a detailed example adapted from Martins (1991). Suppose we have the ATMS network of Figure 9.4. In this network, $n_1$, $n_2$, $n_4$, and $n_5$ are premises and assumed true. The dependency network also reflects the relations that from premise $n_1$ and $n_2$ we support $n_3$, with $n_3$ we support $n_7$, with $n_4$ we support $n_7$, with $n_4$ and $n_5$ we support $n_6$, and finally, with $n_6$ we support $n_7$.

Figure 9.5 presents the subset/superset lattice for the premise dependencies found in Figure 9.4. This lattice of subsets of premises offers a useful way to visualize the space of combinations of premises. Thus, if some premise is found to be suspect, the ATMS will be able to determine how that premise relates to other premise support subsets. For example, node $n_3$ in Figure 9.4 will be supported by all sets of premises that are above $\{n_1,n_2\}$ in the lattice of Figure 9.5.

The ATMS reasoner removes contradictions by removing from the nodes those sets of premises that are discovered to be inconsistent. Suppose, for example, we revise the support for the reasoning reflected by Figure 9.4 to make $n_3$ a contradiction node. Since the label for $n_3$ is $\{n_1, n_2\}$, this set of premises is determined to be inconsistent. When this inconsistency is discovered, all the sets of premises that are in the superset relation to $\{n_1, n_2\}$ in Figure 9.5 are marked as inconsistent (wwe have circled these) and removed from the dependency network. In this situation, one of the possible labellings supporting $n_7$ will also have to be removed. A full description of the contradiction-removal algorithm may be obtained from deKleer (1986).

There are several other important contributions to TMS reasoning. *Logic-based TMS* is based on the work of McAllester (1978). In LTMS, relationships between propositions

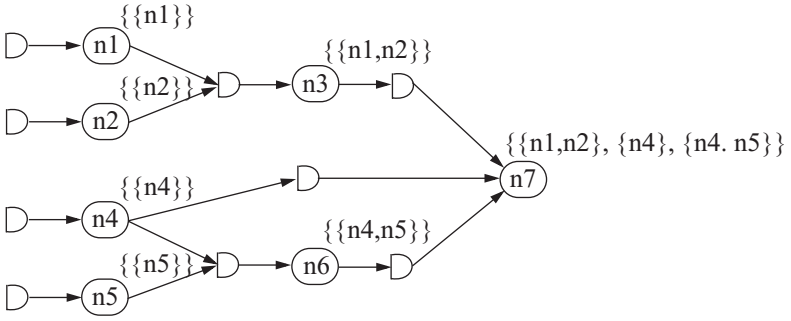


Figure 9.4 An ATMS labeling of nodes in a dependency network.

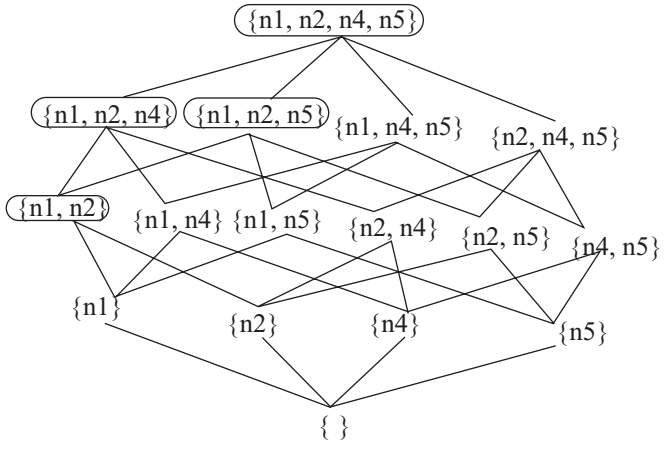


Figure 9.5 The lattice for the premises of the network of Figure 9.4. Circled sets indicate the hierarchy of inconsistencies, after Martins (1991).

are represented by clauses which can be used to deduce the truth values of any of the propositions they describe. Another approach, the *multiple belief reasoner* (MBR) is similar to the ATMS reasoner except that the problem solver and the truth maintenance system are merged into a single system. MBR is based on a logic language called SWM* which describes knowledge states. Each knowledge state is composed of a pair of descriptors, the first reflecting a knowledge base and the second a set of sets of known inconsistent premises within the knowledge base. Algorithms for checking inconsistencies during reasoning may be found in Martins (1991). Further discussion on truth maintanence systems may be found in *The Encyclopedia of Artificial Intelligence* (Shapiro 1992).

### 9.1.3 Logics Based on Minimum Models

In the previous sections, we extended logic by several different modal operators that were specifically designed to reason about the world *as it usually is*, relaxing the requirement that our knowledge of the world be somehow complete. These operators were born of the necessity of creating a more flexible and revisable view of the world. In this section we present logics designed specifically for two situations: first, to reason where a set of assertions specifies only those things that are true, and second, to reason where, because of the nature of a problem-solving task, sets of conjectures are usually true. In the first situation we use the *closed world assumption* and in the second *circumscription*. Both of these approaches to logic are often referred to as *reasoning over minimal models*.

We saw in Section 2.3 that a *model* is an interpretation that satisfies S, a set of predicate expressions, for all variable assignments. There are a number of ways of defining what is meant by a *minimum model.* We define a minimum model as *a model such that there are no smaller models that can satisfy the set of expressions* S *for all variable assignments.*

The idea that makes minimum models important for reasoning is this: there are a (potentially) infinite number of predicates that can be used to describe situations in the world. Consider, for example, the limitless predicates that can be used to describe the situation for the farmer-wolf-goat-cabbage problem (Section 3.5, Exercise 2): the boat is not slowly sinking, the river banks are close enough that rowing will get the boat across, the wind and current are not relevant factors, and so on. When we describe a problem we are usually quite parsimonious in our descriptions. We create only those predicates that are both relevant and needed to solve the problem.

The *closed world assumption* is based on this minimum model of the world. Exactly those predicates that are necessary for a solution are created. The closed world assumption effects the semantics of negation in reasoning. For example, if we wanted to determine whether a student is an enrolled member of a class, we could go to the enrolment database, and if the student is not explicitly listed in that database (the minimal model), he or she would not be enrolled. Similarly, if we wanted to know if two cities were directly connected by a plane flight, we would go to the listing of all airline connections. We would infer, if the direct flight is not listed there (the minimal model), that it does not exist.

The closed world assumption is a statement that if our computational system cannot conclude that p(X) is true, then not(p(X)) must be true. As we will see in Section 14.4, the closed world assumption supports Prolog inferencing. In Section 14.4 we see the three assumptions (axioms) implicit in the use of minimal models. These axioms are the *unique name*, i.e., that all atoms with distinct names are distinct; *the closed world*, i.e., the only instances of a relation are those implied by the clauses present; and *domain closure*. i.e., the atoms of the domain are exactly those of the model. When these three are satisfied, a minimum model becomes a full logic-based specification. If the axioms are not satisfied, some form of a truth maintenance algorithm is required.

If the closed world requires that all the predicates that make up a model be stated, *circumscription* (McCarthy 1980, Lifschitz 1984, McCarthy 1986) requires that *only* those predicates relevant to the problem solving are stated. In circumscription, axioms are added to a system that forces a minimal interpretation on the predicates of the knowledge base.

These *meta-predicates* (predicates about the problem statement's predicates) describe the manner in which particular predicates are to be interpreted. That is, they delimit, or circumscribe, the possible interpretations of predicates.

McCarthy (1980) introduced the idea of circumscription with a thought experiment on the missionaries and cannibals problem. The problem statement asks the solver to devise a series of moves in which six characters, under a set of constraints, can use a boat to cross a river. McCarthy brings up a large number of absurd situations that, quite legitimately, can be asked about the problem statement. A number of these, such as a slowly sinking boat or a wind factor, were presented earlier in this section. Although humans regard these situations as absurd, the reasoning we use to do so is not obvious. The circumscription axioms that McCarthy would add to the problem specification, would precisely delimit the predicates that describe the problem.

As another example of circumscription, consider a predicate expression from an object-oriented common sense reasoning specification*;* see discussion, Section 9.4:

$$\forall\ X\ \text{bird}(X) \wedge \text{not}\ (\text{abnormal}(X)) \rightarrow \text{flies}(X)$$

This expression might occur in reasoning where one of the properties of *bird* is *flies*. But what could possibly limit the definition of the predicate *abnormal*? That the bird is not a penguin, that it does not have a broken wing, that it is not dead? The specification of the predicate *abnormal* is potentially undecidable.

Circumscription uses an axiom schema, or set of meta rules, within first-order predicate calculus to generate predicates for the problem domain. The schema rules cause certain formulae to have the smallest possible extensions. For example, if B is a belief system including world knowledge K and domain knowledge A(p) about a predicate p, then we may consider p to be minimized, in that as few atoms $a_i$ as possible satisfy $p(a_i)$ as is consistent with A(p) and K. The world knowledge K together with A(p) and the circumscription schema are used to derive conclusions in standard first-order predicate calculus. These conclusions are then added to B, the belief system.

As an example, suppose in the blocks world, Section 8.4, we have the expression:

$$\text{isblock}(a) \wedge \text{isblock}(b) \wedge \text{isblock}(c)$$

asserting that a, b, and c are blocks. Circumscribing the predicate isblock gives:

$$\forall X\ (\text{isblock}(X) \leftarrow ((X = a)\ v\ (X = b)\ v\ (X = c)))$$

This expression asserts that only blocks a, b, and c, i.e., just those objects that the isblock predicate requires, are blocks in this domain. Similarly, the predicate:

$$\text{isblock}(A)\ v\ \text{isblock}(B)$$

can be circumscribed to:

$$\forall X\ (\text{isblock}(X) \leftarrow ((X = a)\ v\ (X = b)))$$

For full details, including the schema axioms used to derive these results, see McCarthy (1980, Section 4).

Circumscription, when used with operators such as abnormal, is much like the closed world assumption in that it produces exactly those variable bindings that abnormal can support. The circumscription algebra, however, allows us to extend this reasoning across predicate representations in that, as we just noted, if we have the predicate p(X) v q(X), we may circumscribe either predicate p or q or both. Thus, unlike the closed world assumption, circumscription allows us to describe the instantiations possible across sets of predicate descriptions.

Further research in circumscriptive logics may be found in Genesereth and Nilsson (1987). Lifschitz (1986) has made an important contribution by proposing a point-wise circumscription in which the minimum model can be carried out for particular predicates and their possible instantiations, rather than for the full domain. Another important contribution is that of Perlis (1988) where reasoning can be about a particular agent's lack of knowledge. For further discussion see also (Shapiro 1992).

### 9.1.4 Set Cover and Logic-Based Abduction (Stern 1996)

As noted in the introduction of this chapter, in abductive reasoning, we have rules of the form $p \rightarrow q$, along with a reasonable belief in q. We wish then to make a case for the truth of predicate p. Abductive reasoning is not sound, but what is often called *reasoning to the best explanation* for the presence of the data q. In this section, we look more closely at the generation of explanations in domains of abductive inference.

In addition to the accounts of abductive reasoning already presented, AI researchers have also used set cover and logic supported analyses. The *set cover* approach to abduction attempts to explain the act of adopting a revocable belief in some explanatory hypothesis on the grounds that it explains an otherwise unexplainable set of facts. The *logic-based* approach to abduction describes inference rules for abduction along with a definition of their legitimate form(s) for use.

The *set cover* approach, defines an abductive explanation as a covering of predicates describing observations by predicates describing hypotheses. Reggia et al. (1983) describe a cover based on a binary causal relation R where R is a subset of {Hypotheses X Observations}. Thus, an abductive explanation of a set of observations S2 is another set of hypotheses S1 sufficient to cause S2. An optimal explanation according to the set cover approach, is the minimal set cover of S2. The weakness of this approach is that it reduces explanation to a simple list of causal hypotheses (from S1). In situations where there are interrelated or interacting causes or where an understanding of the structure or sequencing of causal interactions is required, the set cover model is inadequate.

Logic-based approaches to abduction on the other hand, rest on a more sophisticated notion of explanation. Levesque (1989) defines an abductive explanation of some previously unexplained set of observations O as a minimal set of hypotheses H consistent with an agent's background knowledge K. The hypotheses H together with the background knowledge K must entail O. More formally:

abduce(K, O) = H, if and only if

1. K does not entail O
2. H ∪ K entails O
3. H ∪ K is consistent, and
4. No subset of H has properties 1, 2, and 3.

Note that in general many sets of hypotheses may exist; that is, there may be many potential abductive sets of explanations for a given set of observations O.

The logic-based definition of abductive explanation suggests a corresponding mechanism for explanation discovery in the context of a knowledge-based system. If the explanatory hypotheses must entail the observations O, then the way to construct a complete explanation is to reason backwards from O. As we saw in Sections 3.3 and 8.2, we may start from the conjunctive components of O and reason back from consequents to their antecedents.

This backchaining approach also seems natural because the conditionals which support the backchaining can readily be thought of as causal laws, thus capturing the pivotal role which causal knowledge plays in the construction of explanations. The model is also convenient because it fits nicely something with which the AI community already has experience: backchaining and computational models for deduction.

There are also clever ways of finding the complete set of abductive explanations. Assumption-based truth-maintenance systems ATMS (deKleer 1986, Section 9.2.3), contain an algorithm for computing minimal support sets, the set of (non-axiom) propositions that logically entail a given proposition in a theory. To find all possible abductive explanations for a set of observations, we merely take the Cartesian product over the support sets.

As simple, precise, and convenient as the logic-based account of abduction is, there are two related shortcomings: high computational complexity and semantic weakness. Selman and Levesque (1990) found the complexity of abduction tasks similar to that involved in computing support sets for an ATMS. The standard proof that the ATMS problem is NP-hard depends on the existence of problem instances with an exponential number of solutions. Selman and Levesque avoid the number of potential solutions complexity issue by asking whether finding a smaller set of solutions is also NP-hard. Given a Horn clause knowledge base, see Section 14.2, Selman and Levesque produce an algorithm that finds a single explanation in order O(k*n) where k indicates the number of propositional variables and n the number of occurrences of literals. However, when restrictions are placed on the kinds of explanations sought, the problem again becomes NP-hard, even for Horn clauses.

One interesting result from the Selman and Levesque (1990) analysis is the fact that adding certain kinds of goals or restrictions to the abduction task actually makes computation significantly harder. From the naive viewpoint of the human problem solver, this added complexity is surprising: the human assumes that the addition of further constraints to the search for relevant explanations makes the task easier. The reason the abduction task is harder in the logic-based model is that it only contributes additional clauses to the problem, not additional structure useful for deriving a solution.

Explanation discovery in the logic-based model is characterized as the task of finding a set of hypotheses with certain logical properties. These properties, including consistency with the background knowledge and entailment of what is to be explained, are meant to capture the *necessary* conditions of explanations: the minimal conditions which a set of explanatory hypotheses must satisfy in order to count as an abductive explanation. Proponents of this approach believe that by adding additional constraints, the approach can be extended to provide a characterization of good or reasonable explanations.

One simple strategy for producing quality explanations is to define a set of fact clauses that are abducible, that is, from which candidate hypotheses must be chosen. This clause set allows search to be restricted in advance to those factors that can potentially play a causal role in the chosen domain. Another strategy is to add selection criteria for evaluating and choosing between explanations. Various selection criteria have been proposed, including *set minimality*, which prefers one hypothesis set over another, where both are consistent and entail what is to be explained, if the first is contained in the second. A *simplicity* criterion gives preference to parsimonious hypothesis sets, those containing fewer unverified assumptions (Levesque 1989).

Both minimality and simplicity can be seen as applications of Occam's razor. Unfortunately, set minimality is of limited power as a search pruning tool; it only eliminates final explanations which are supersets of existing explanations. Simplicity alone is also of questionable validity as a search selection criterion. It is not difficult to construct examples in which an explanation requiring a larger hypothesis set is preferable to some simpler but shallower set of hypotheses. Indeed, complex causal mechanisms will usually require larger hypothesis sets; however, the abduction of such causal mechanisms may well be justified, particularly when the presence of certain key elements of that mechanism have already been verified by observation.

Two other mechanisms for explanation selection are also interesting because they take into account both properties of the hypothesis set as well as properties of the proof procedure. First, *cost-based abduction* places a cost on potential hypotheses as well as a cost on rules. The total cost of the explanation is computed on the basis of the total cost of the hypotheses plus the cost of the rules used to abduce the hypotheses. Competing hypothesis sets are then compared according to cost. One natural heuristic that can be attached to this scheme is the probabilistic one (Charniak and Shimony 1990). Higher costs for hypotheses represent less likely events; higher costs for rules represent less probable causal mechanisms. Cost-based metrics can be combined with least-cost search algorithms, such as best-first search, Chapter 4, considerably reducing the computational complexity of the task.

A second mechanism, *coherence-based selection,* is particularly appealing when what is to be explained is not a simple proposition but rather a set of propositions. Ng and Mooney (1990) have argued that a coherence metric is superior to a simplicity metric for choosing explanations in the analysis of natural language text. They define coherence as a property of a proof graph where explanations with more connections between any pair of observations and fewer disjoint partitions are more coherent. The coherence criterion is based on the heuristic assumption that what we are asked to explain is a single event or action with multiple aspects. The justification for a coherence metric in natural language understanding is based on Gricean felicity conditions, that is, the speaker's obligation to

be coherent and pertinent (Grice 1975). It is not difficult to extend their argument to a variety of other situations. For example, in diagnosis, the observations which comprise the initial set of things to be explained are brought together because they are believed to be related to the same underlying fault or failure mechanism.

In Section 9.1 we considered extensions to traditional logic that supported reasoning with uncertain or missing data. We next describe non-logic alternatives for reasoning in situations of uncertainty, including the Stanford certainty factor algebra, reasoning with fuzzy sets, and the Dempster-Shafer theory of evidence.

## 9.2 Abduction: Alternatives to Logic

The logic-based approaches of Section 9.1 are cumbersome and computationally intractable for many applications, especially expert systems. Alternatively, several early expert system projects, e.g., PROSPECTOR, attempted to adapt Bayesian techniques, Section 9.3, for abductive inference. The independence assumptions, continuous updates of statistical data, and the calculations required to support stochastic inference algorithms limits this approach. An alternative inference mechanism, designed to aliminate these constraints, was used at Stanford for the development of early expert systems, including MYCIN (Buchanan and Shortliffe 1984).

When reasoning with heuristic knowledge, human experts are able to give adequate and useful estimates of their confidences in rule relationships. Humans weight conclusions with terms like *highly probable*, *unlikely*, *almost certainly*, or *possible*. These weights are clearly not based in careful analysis of probabilities. Instead, they are themselves heuristics derived from experience in reasoning about the problem domain. In Section 9.2 we introduce three methodologies for abductive inference: Stanford certainty theory, fuzzy reasoning, and the Dempster–Shafer theory of evidence. In Section 9.3 we present stochastic approaches to uncertainty.

### 9.2.1 The Stanford Certainty Factor Algebra

*Stanford certainty theory* is based on a number of observations. The first is that in traditional probability theory, the sum of confidence for a relationship and confidence against the same relationship must add to one. However, it is often the case that a human expert might have confidence 0.7 (of 1.0) 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”.

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 have been shown to have little effect on the overall running system. This role of certainty measures also confirms the belief that "the knowledge gives the power," that is, the integrity of the knowledge itself best supports the production of correct diagnoses.

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, given those premises. 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 added 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 $(0.7) \times (0.4) = 0.28$.
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 $(0.3) \times (0.4) = 0.12$.

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) × CF(R2)) when CF(R1) and CF(R2) are positive,
CF(R1) + CF(R2) + (CF(R1) × CF(R2)) when CF(R1) and CF(R2) are negative,

and

$$\frac{CF(R1) + CF(R2)}{1 - MIN(|CF(R1)|, |CF(R2)|)}$$

otherwise, where $|X|$ is the absolute value of X.

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 5.4, 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(D|A,B), etc. 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 search space. 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, 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.

### 9.2.2 Reasoning with Fuzzy Sets

There are two assumptions that are essential for the use of formal set theory. The first is with respect to set membership: for any element and a set belonging to some universe, the element is either a member of the set or else it is a member of the complement of that set. The second assumption, referred to as *the law of excluded middle*, states that an element cannot belong to both a set and also to its complement. Both these assumptions are violated in Lotfi Zadeh's *fuzzy set theory.* In fact, the sets and reasoning laws of traditional set theory are referred to as *crisp*, from the fuzzy set viewpoint.

Zadeh's main contention (Zadeh 1983) is that, although probability theory is appropriate for measuring randomness of information, it is inappropriate for measuring the *meaning* of information. Indeed, much of the confusion surrounding the use of English words and phrases is related to lack of clarity (vagueness) rather than randomness. This is a crucial point for analyzing language structures and can also be important in creating a measure of confidence in production rules. Zadeh proposes *possibility theory* as a measure of vagueness, just as probability theory measures randomness.

Zadeh's theory expresses lack of precision in a quantitative fashion by introducing a set membership function that can take on real values between 0 and 1. This notion of a *fuzzy set* can be described as follows: let S be a set and s a member of that set. A fuzzy subset F of S is defined by a membership function mF(s) that measures the "degree" to which s belongs to F.

A standard example of a fuzzy set, as presented in Figure 9.6, is for S to be the set of positive integers and F to be the fuzzy subset of S called *small* integers. Now various integer values can have a "possibility" distribution defining their "fuzzy membership" in the set of small integers: mF (1) = 1.0, mF (2) = 1.0, mF (3) = 0.9, mF(4) = 0.8, ... , mF (50) = 0.001, etc. For the statement that positive integer X is a *small integer*, mF creates a possibility distribution across all the positive integers (S).

Fuzzy set theory is not concerned with how these possibility distributions are created,

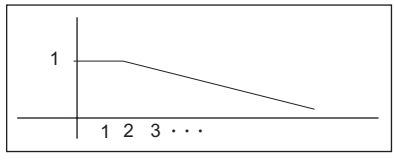


Figure 9.6 The fuzzy set representation for small integers.

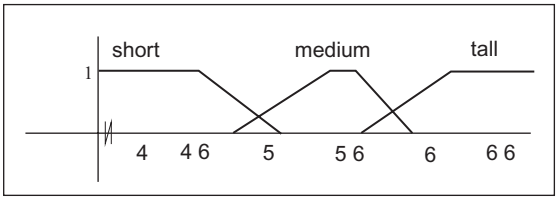


Figure 9.7 A fuzzy set representation for the sets short, median, and tall males.

but rather with the rules for computing the combined possibilities over expressions that contain fuzzy variables. Thus, it includes rules for combining possibility measures for expressions containing fuzzy variables; in fact, the laws for the or, and, and not of these expressions are similar to those just presented for the Stanford certainty factor algebra; see Section 9.2.1.

For the fuzzy set representation of the set of small integers, Figure 9.6, each integer belongs to this set with an associated confidence measure. In the traditional logic of "crisp" sets, the confidence of an element being in a set must be either 1 or 0. Figure 9.7 offers a set membership function for the concept of *short*, *medium*, and *tall* male humans. Note that any one person can belong to more than one set, for example, a 5' 9" male belongs to both the set of *medium* as well as to the set of *tall* males.

We next demonstrate rules for combining and propagating fuzzy measures by presenting part of a problem, now classic in the fuzzy set literature, a control regime for an inverted pendulum. Figure 9.8 presents a pendulum, inverted, which we desire to keep in balance and pointing upward. We keep the pendulum in balance by moving the base of the system to offset the force of gravity acting on the pendulum. There are sets of differential equations that can deterministically keep the pendulum in equilibrium (Ross 1995). The

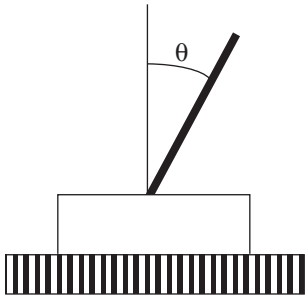


Figure 9.8 The inverted pendulum and the angle θ and dθ/dt input values.

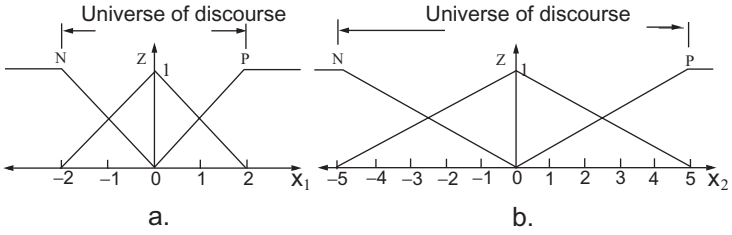


Figure 9.9 The fuzzy regions for the input values θ (a) and dθ/dt (b).

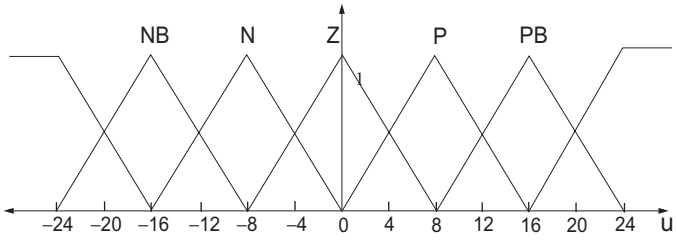


Figure 9.10 The fuzzy regions of the output value u, indicating the movement of the pendulum base.

advantage of the fuzzy approach to controlling this pendulum system is that an algorithm may be established to control the system efficiently and in real time. We next describe this control regime.

We simplify the pendulum problem by presenting it in two dimensions. These two measurements are used as input values to the controller, as may be seen in Figure 9.9: First, the angle θ, the deviation of the pendulum from the vertical, and second, the speed, dθ/dt, at which the pendulum is moving. Both these measures are positive in the quadrant to the right of vertical and negative to the left. These two values are given to the fuzzy controller at each iteration of the system. The output of the controller is a movement and a direction for the base of the system, instructions that are intended to keep the pendulum in balance.

To clarify the actions of the fuzzy controller we describe the fuzzy set solution process. Data describing the state of the pendulum, θ and dθ/dt, are interpreted as fuzzy measures, see Figure 9.9, and presented to a fuzzy rule set. This step is often made very efficient by use of a structure called a *fuzzy associative matrix* or FAM, Figure 9.12, where input/output relations are directly encoded. Rules are not chained together as in traditional rule-based problem solving. Rather, all matched rules fire and then their results are combined. This result, usually represented by an area of the fuzzy output parameter space, Figure 9.10, is then *defuzzified* to return the control response. Note that both the original input and eventual output of the controller are crisp values. These are exact readings of some monitor, the inputs, and precise instructions for control actions, the output.

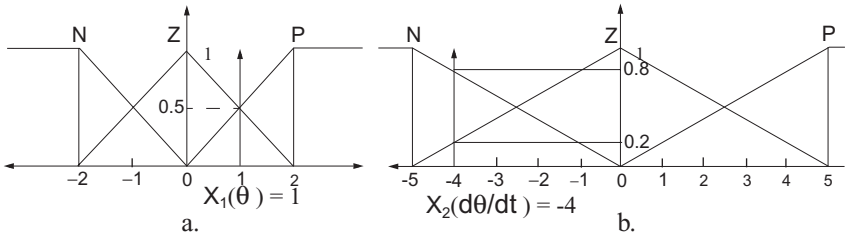


Figure 9.11 The fuzzification of the input measures $x_1$=1, $x_2$ = -4.

| $x_1$ \ $x_2$ | P | Z | N |
|---|---|---|---|
| P | PB | P | Z |
| Z | P | Z | N |
| N | Z | N | NB |

Figure 9.12 The Fuzzy Associative Matrix (FAM) for the pendulum problem. The input values are on the left and top.

We next describe the fuzzy regions for the input values, θ and dθ/dt. This example simplifies the situation, for example, in the number of fuzzy regions of input values, but shows the full cycle of rule application and the response of the controller. The input value θ is partitioned into three regions, Negative, Zero, and Positive, where θ ranges between –2 and +2 radians, as may be seen in Figure 9.9a. Figure 9.9b represents the three regions into which the second input value, dθ/dt, is partitioned, again Negative, Zero, and Positive, ranging between –5 and +5 degrees per second.

Figure 9.10 represents the partitioning of the output space, where we use the middle five regions, Negative Big, Negative, Zero, Positive, and Positive Big. The measure, between –24 and +24 represents the movement and direction of each response.

Suppose the simulation begins and the first values given to the controller are θ = 1 and dθ/dt = –4. Figure 9.11 reflects the fuzzification of these input measures. In each situation, the input value impacts two regions of the fuzzy input space. For θ, the values are Zero, with 0.5, and Positive, with 0.5 possibility measures. For dθ/dt, they are Negative with 0.8, and Zero with 0.2 possibility measures.

Figure 9.12 presents a simplified form of the fuzzy associative matrix for this problem. The input values to the table for θ, or $x_1$, are down the left side, and for dθ/dt, or $x_2$,

are across the top of the matrix. The $3 \times 3$ table of the lower right corner of the FAM then gives the output values. For example, if the θ is Positive, and dθ/dt is Negative, the FAM returns the value of Zero movement of the pendulum system. Note that the response still must be defuzzified using the Zero output region of Figure 9.10.

In this case, because each input value touched on two regions of the input space, four rules must be applied. As noted above, the combination rules for fuzzy systems are similar to those of the Stanford certainty factor algebra. In fact, Zadeh (Buchanan and Shortliffe 1984) was the first (historically) to propose these combination rules for the algebra of fuzzy reasoning. If the measures of two premises are ANDed together, the minimum of their measures is taken as the measure of the rule. If two premises are ORed, the maximum of their measures is taken.

In our example, all the premise pairs are ANDed together, so the minimum of their measures is taken as the measure of the rule result:

IF $x_1$ = P AND $x_2$ = Z THEN u = P
min(0.5, 0.2) = 0.2 P

IF $x_1$ = P AND $x_2$ = N THEN u = Z
min(0.5, 0.8) = 0.5 Z

IF $x_1$ = Z AND $x_2$ = Z THEN u = Z
min(0.5, 0.2) = 0.2 Z

IF $x_1$ = Z AND $x_2$ = N THEN u = N
min(0.5, 0.8) = 0.5 N

Next the output results are combined. In this example, we union together the two areas of Figure 9.10, indicated by the results of this set of two rules firing. There are a number of possible defuzzification techniques (Ross 1995). We have choosen one of the commonest, the *centroid method*. To use this method the centroid of the union of the areas of the output values becomes the final value the controller applies to the pendulum. The union, as well as the centroid for the union, are presented in Figure 9.13. After this output or result is applied to the system, θ and dθ/dt are sampled again and the control cycle is repeated.

There are a number of issues we have not addressed in describing fuzzy reasoning systems, including patterns of oscillations within the convergence process and optimum sampling rates. Fuzzy systems, especially in the area of control, offer engineers a powerful and efficient tool for dealing with imprecision in measurement.

### 9.2.3 The Dempster–Shafer Theory of Evidence

To this point in our discussion of reasoning under uncertainty, we described techniques which consider individual propositions and assign to each a causal influence or a numeric estimate of the degree of belief that we might have, given sets of evidence. One of the limitations of probabilistic approaches to uncertainty is their use of a single quantity to measure what may be a very complex situation. Often, uncertainty results from a combination

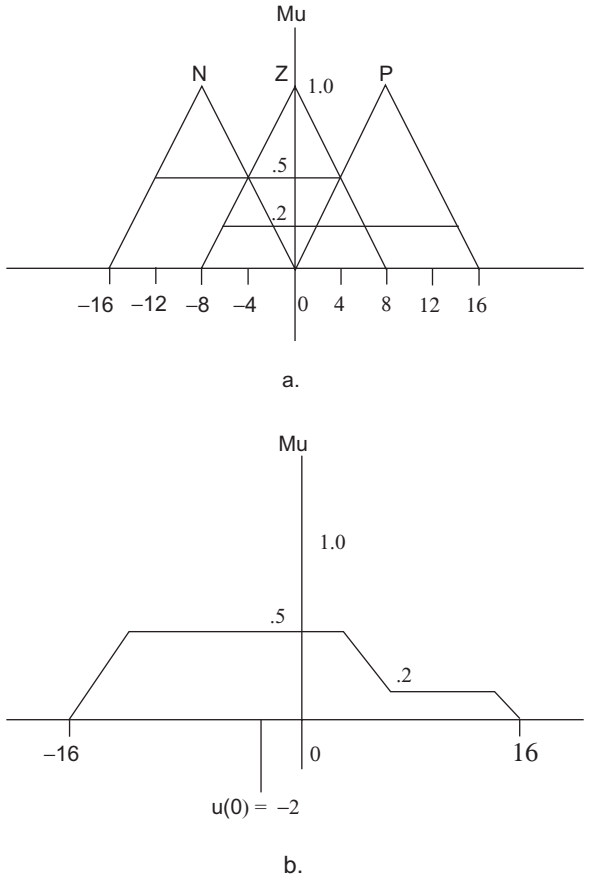


Figure 9.13 The fuzzy consequents (a) and their union (b). The centroid of the union (–2) is the crisp output.

of missing evidence, the inherent limitations of heuristic rules, and the limitations of our own knowledge.

An alternative approach, called the *Dempster–Shafer theory of evidence,* considers sets of propositions and assigns to each of them an interval [belief, plausibility] within which the degree of belief for each proposition must lie. This *belief measure*, denoted bel, ranges from zero, indicating no evidence of support for a set of propositions, to one, denoting certainty. The *plausibility* of a proposition p, pl(p), is defined:

$$pl(p) = 1 - bel(not\ (p))$$

Thus, plausibility also ranges between zero and one and reflects how evidence of not (p) relates to the possibility for belief in p. If we have certain evidence of not (p) then

bel(not(p)) will be one and pl(p) will be zero. The only possible value for bel(p) is also zero.

Suppose we have two competing hypotheses $h_1$ and $h_2$. When we have no information supporting either hypothesis, they each have the belief/plausibility range [0,1]. As evidence is gathered, we expect these intervals to shrink, representing the increased confidence for the hypotheses. In a Bayesian domain, we would probably begin (with no evidence) by distributing the prior probabilities equally among the two hypotheses, giving each $p(h_i) = 0.5$, say. Dempster–Shafer makes it clear that we have no evidence when we start; the Bayesian approach, on the other hand, can result in the same probability measure no matter how much data we have. Thus, Dempster–Shafer can be very useful when it is important to make a decision based on the amount of evidence that has been collected.

To summarize, Dempster and Shafer address the problem of measuring certainty by making a fundamental distinction between lack of certainty and ignorance. In probability theory we are forced to express the extent of our knowledge about an hypothesis h in a single number, p(h). The problem with this approach, say Dempster and Shafer, is that we simply cannot always know the values of its supporting probabilities, and therefore, any particular choice of p(h) may not be justified.

The Dempster–Shafer belief functions satisfy axioms that are weaker than those of probability theory, that is, it reduces to probability theory when all probabilities are obtainable. Belief functions allow us to use our knowledge to bound the assignment of probabilities to events in the absence of exact probabilities.

The Dempster–Shafer theory is based on two ideas: first, the idea of obtaining degrees of belief for one question from subjective probabilities for related questions, and second, the use of a rule for combining these degrees of belief when they are based on independent items of evidence. This combination rule was originally proposed by Dempster (1968). Next, we present an informal example of Dempster–Shafer reasoning, then present Dempster's rule, and finally, apply that rule to a more realistic situation.

Suppose I have subjective probabilities for the reliability of my friend Melissa. The probability that she is reliable is 0.9, and that she is unreliable, 0.1. Suppose Melissa tells me that my computer was broken into. This statement is true if Melissa is reliable, but it is not necessarily false if she is unreliable. So Melissa's testimony alone justifies a degree of belief of 0.9 that my computer was broken into and a 0.0 belief that it was not. Belief of 0.0 does not mean that I am sure that my computer was not broken into, as a probability of 0.0 would. It merely means that Melissa's testimony gives me no reason to believe that my computer was not broken into. The plausibility measure, pl, in this situation is:

pl(computer_broken_into) = 1 – bel(not (computer_broken_into)) = 1 – 0.0

or 1.0, and my belief measure for Melissa's testimony is [0.9 1.0]. Note that there is still no evidence that my computer was not broken into.

We next consider Dempster's rule for combining evidence. Suppose my friend Bill also tells me that my computer was broken into. Suppose the probability that Bill is reliable is 0.8 and that he is unreliable is 0.2. I also must suppose that Bill's and Melissa's testimonies about my computer are independent of each other; that is, they have separate reasons for telling me that they think my computer was broken into. The event that Bill is

reliable must also be independent of Melissa's reliability. The probability that both Bill and Melissa are reliable is the product of their reliabilities, or 0.72; the probability that they both are unreliable is the product 0.02. The probability that at least one of the two is reliable is 1 − 0.02, or 0.98. Since they both said that my computer was broken into and there is a probability of 0.98 that at least one of them is reliable, I will assign to the event of my computer being broken into a [0.98 1.0] degree of belief.

Suppose that Bill and Melissa disagree on whether my computer was broken into: Melissa says that it was, and Bill says that it was not. In this case, they cannot both be correct and they cannot both be reliable. Either both are unreliable or only one is reliable. The prior probability that only Melissa is reliable is $0.9 \times (1 - 0.8) = 0.18$, that only Bill is reliable is $0.8 \times (1 - 0.9) = 0.08$, and that neither is reliable is $0.2 \times 0.1 = 0.02$. Given that at least one is not reliable, $(0.18 + 0.08 + 0.02) = 0.28$, we can also compute the posterior probability that only Melissa is reliable as $0.18 / 0.28 = 0.643$ and my computer was broken into, or the posterior probability that only Bill was right, $0.08 / 0.28 = 0.286$, and my computer was not broken into.

We have just used the Dempster rule to combine beliefs. When Melissa and Bill both reported the computer break-in, we summed the three hypothetical situations that supported the break-in: Bill and Melissa are both reliable; Bill is reliable and Melissa not; and Melissa is reliable and Bill not. The belief, 0.98, was the sum of these possible supporting hypothetical scenarios. In the second use of the Dempster rule, the witnesses disagreed. Again we summed all possible scenarios. The only impossible situation was that they were both reliable; thus either Melissa was reliable and Bill not, Bill was reliable and Melissa not, or neither was reliable. The sum of these three gives a belief of break-in of 0.64. The belief that my computer was not broken into (Bill's opinion) was 0.286; since the plausibility of break-in is $1 - \text{bel}(\text{not}(\text{break in}))$ or 0.714, the belief measure is [0.28 0.714].

To use the Dempster rule, we obtain degrees of belief for one question (Was my computer broken into?) from probabilities for another question (Are the witnesses reliable?). The rule begins with the assumption that the questions for which we have probabilities are independent, but that this independence is only a priori. It disappears when we have conflict between the different items of evidence.

Using the Dempster–Shafer approach in a specific situation involves solving two related problems. First, we sort the uncertainties of the situation into a priori independent pieces of evidence. Second, we carry out Dempster's rule. These two tasks are related: Suppose, again, that Bill and Melissa told me, independently, that they believed my computer was broken into. Suppose also that I had called a repair person to check my computer, and that both Bill and Melissa had witnessed this. Because of this common event, I can no longer compare degrees of belief. However, if I consider explicitly the possibility of the repair person's working on my computer, then I have three independent items of evidence: Melissa's reliability, Bill's reliability, and evidence for the presence of the repair person, which I can then combine with Dempster's rule.

Suppose we have an exhaustive set of mutually exclusive hypotheses we call $\Omega$. Our goal is to attach some measure of belief, $m$, to the various subsets $Z$ of $\Omega$; $m$ is sometimes called the *probability density function* for a subset of $\Omega$. Realistically, not all evidence is directly supportive of individual elements of $\Omega$. In fact, evidence most often supports different subsets $Z$ of $\Omega$. In addition, since the elements of $\Omega$ are assumed to be mutually

exclusive, evidence in favor of some may have an effect on our belief in others. In a purely Bayesian system, Section 9.3, we address both of these situations by listing all the combinations of conditional probabilities. In the Dempster–Shafer system we handle these interactions by directly manipulating the sets of hypotheses. The quantity $m_n(Z)$ measures the amount of belief that is assigned to the subset $Z$ of hypotheses, and $n$ represents the number of sources of evidence.

Dempster's rule states:

$$m_n(Z) = \frac{\sum_{X \cap Y = Z} m_{n-2}(X) m_{n-1}(Y)}{1 - \sum_{X \cap Y = \phi} m_{n-2}(X) m_{n-1}(Y)}$$

For example, the belief in an hypothesis $Z$, with $n = 3$ sources of evidence, $m_3(Z)$, is the sum of the products of the hypothetical situations, $m_1(X)$ and $m_2(Y)$, whose co-occurrence supports $Z$, that is, $X \cap Y = Z$. The denominator of Dempster's rule acknowledges, as we see in the following example, that $X$ and $Y$ can have an empty intersection, and the sum of the confidences must be normalized by one minus the sum of these values.

We next apply Dempster's rule to a situation of medical diagnosis. Suppose $Q$ represents the domain of our focus, containing four hypotheses: that a patient has a cold ($C$), flu ($F$), migraine headaches ($H$), or meningitis ($M$). Our task is to associate measures of belief with hypothesis sets within $Q$. As just noted, these are *hypothesis sets* since evidence need not support individual hypotheses exclusively. For example, having a fever could support $\{C,F,M\}$. Since the elements of $Q$ are treated as mutually exclusive hypotheses, evidence in favor of some may affect belief in others. As already noted, the Dempster–Shafer approach addresses interactions by handling the sets of hypotheses directly.

For the probability density function, $m$, and all subsets $Z$ of the set $Q$, the quantity $m(q_i)$ represents the belief that is currently assigned to each $q_i$ of $Q$ with the sum of all the $m(q_i)$ equal to one. If $Q$ contains $n$ elements, then there are $2^n$ subsets of $Q$. Even though addressing $2^n$ values may appear daunting, it usually turns out that many of the subsets will never occur. Thus, there is some simplification of the solution process since these values can be ignored, because they have no utility in the problem domain. Finally, the plausibility of $Q$ is $pl(Q) = 1 - \Sigma\, m(q_i)$, where the $q_i$ are the sets of hypotheses that have some supporting belief. If we have no information about any hypotheses, as is often the case when we start a diagnosis, then $pl(Q) = 1.0$.

Suppose the first piece of evidence is that our patient has a fever, and that this supports $\{C,F,M\}$ at 0.6. We call this first belief $m_1$. If this is our only hypothesis, then $m_1\{C,F,M\} = 0.6$, where $m_1(Q) = 0.4$, to account for the remaining distribution of belief. It is crucial to note that $m_1(Q) = 0.4$ represents the remainder of our belief distribution, that is, all other possible beliefs across $Q$ and not our belief in the complement of $\{C,F,M\}$.

Suppose that we now acquire some new data for our diagnosis, say that the patient has extreme nausea, which suggests $\{C,F,H\}$ with support level 0.7. For this belief, call it $m_2$, we have $m_2\{C,F,H\} = 0.7$ and $m_2(Q) = 0.3$. We use Dempster's rule to combine these two beliefs, $m_1$ and $m_2$. Let $X$ be the set of subsets of $Q$ to which $m_1$ assigns a nonzero value and $Y$ be the set of subsets of $Q$ to which $m_2$ assigns a nonzero value. We then create a combination belief, $m_3$ defined on subsets $Z$ of $Q$ by using Dempster's rule.

In applying Dempster's rule to the diagnoses, first note that there are no sets $X \cap Y$ that are empty, so the denominator is 1. The belief distribution for $m_3$ is seen in Table 9.1.

| $m_1$ | $m_2$ | $m_3$ |
|---|---|---|
| $m_1\{C,F,M\} = 0.6$ | $m_2\{C,F,H\} = 0.7$ | $m_3\{C,F\} = 0.42$ |
| $m_1(Q) = 0.4$ | $m_2\{C,F,H\} = 0.7$ | $m_3\{C,F,H\} = 0.28$ |
| $m_1\{C,F,M\} = 0.6$ | $m_2(Q) = 0.3$ | $m_3\{C,F,M\} = 0.18$ |
| $m_1(Q) = 0.4$ | $m_2(Q) = 0.3$ | $m_3(Q) = 0.12$ |

Table 9.1 Using Dempster s rule to obtain a belief distribution for $m_3$.

Using Dempster's rule, the four sets $Z$, all possible ways of intersecting $X$ and $Y$, make up the rightmost column of Table 9.1. Their belief level is computed by multiplying the beliefs for the corresponding elements of $X$ and $Y$ under $m_1$ and $m_2$ respectively. Note also that, in this example, each set in $Z$ is unique, which is often not the case.

We extend our example one final time to show how empty belief sets are factored into the analysis. Suppose we have a new fact, the results of a lab culture that are associated with meningitis. We now have $m_4\{M\} = 0.8$ and $m_4(Q) = 0.2$. We may use Dempster's formula to combine $m_3$, the results of our previous analysis, with $m_4$ to get $m_5$, as can be seen in Table 9.2.

| $m_3$ | $m_4$ | $m_5$ (without denominator) |
|---|---|---|
| $m_3\{C,F\} = 0.42$ | $m_4\{M\} = 0.8$ | $m_5\{\ \} = 0.336$ |
| $m_3(Q) = 0.12$ | $m_4\{M\} = 0.8$ | $m_5\{M\} = 0.096$ |
| $m_3\{C,F\} = 0.42$ | $m_4(Q) = 0.2$ | $m_5\{C,F\} = 0.084$ |
| $m_3(Q) = 0.12$ | $m_4(Q) = 0.2$ | $m_5(Q) = 0.024$ |
| $m_3\{C,F,H\} = 0.28$ | $m_4\{M\} = 0.8$ | $m_5\{\ \} = 0.224$ |
| $m_3(C,F,M) = 0.18$ | $m_4\{M\} = 0.8$ | $m_5\{M\} = 0.144$ |
| $m_3\{C,F,H\} = 0.28$ | $m_4(Q) = 0.2$ | $m_5\{C,F,H\} = 0.056$ |
| $m_3(C.F,M) = 0.18$ | $m_4(Q) = 0.2$ | $m_5\{C,F,M\} = 0.036$ |

Table 9.2 Using Dempster s rule to combine $m_3$ and $m_4$ to get $m_5$.

First, note that $m_5\{M\}$ is produced by the intersections of two different pairs of sets, so the total $m_5\{M\} = 0.240$. We also have the case where several set intersections produce the empty set, $\{\}$. Thus the denominator for Dempster's equation is $1 - (0.336 + 0.224) = 1 - 0.56 = 0.44$. The final combined belief function for $m_5$ is:

| | | |
|---|---|---|
| $m_5\{M\} = 0.545$ | $m_5\{C,F\} = 0.191$ | $m_5\{\ \} = 0.56$ |
| $m_5\{C,F,H\} = 0.127$ | $m_5\{C,F,M\} = 0.082$ | $m_5(Q) = 0.055$ |

Three final comments. First, a large belief assigned to the empty set, as in this final $m_5\{\ \} = 0.56$, indicates that there is conflicting evidence within the belief sets $m_i$. In fact, we designed our example to show several features of Dempster–Shafer reasoning, and, as a consequence, sacrificed medical integrity. Second, when there are large hypothesis sets as well as complex sets of evidence, the calculations for belief sets can get cumbersome, even though, as pointed out earlier, the amount of computation is still considerably less than that for Bayesian reasoning. Finally, the Dempster–Shafer approach is a very useful tool when the stronger Bayesian assumptions may not be justified.

Dempster–Shafer is an example of an algebra supporting the use of subjective probabilities in reasoning. It is sometimes felt that these subjective probabilities better reflect human expert reasoning. In the final section of Chapter 9 we consider further reasoning techniques based on extensions of Bayes' rule, first introduced in Section 5.3.

## 9.3 The Stochastic Approach to Uncertainty

Using probability theory, we can determine, often from a priori argument, the chances of events occurring. We can also describe how combinations of events are able to influence each other. Although the final touches on probability theory awaited the mathematicians of the early twentieth century, including Fisher, Neyman, and Pearson, the attempt to create a combinatorial algebra goes back through the middle ages to the Greeks, including Llull, Porphyry, and Plato (Glymour et al. 1995*a*). The insight supporting probability theory is that if we can understand the frequency with which events have occurred in the past we can use this information (as an inductive bias) to interpret and reason about present data.

In Chapter 5, we noted that there are a number of situations when probabilistic analysis is appropriate. For example, when the world is genuinely random, as in playing a game with well-shuffled cards, or spinning a fair roulette wheel. Further, when many events in the world may not be truly random, but it is impossible to know and measure all their causes and interactions well enough to predict events; statistical correlations are a useful support for interpreting the world. A further role for statistics is as a basis for automated induction and machine learning (for example, the ID3 algorithm of Section 10.3). Finally, recent work has attempted to directly link the notions of probability and causality (Glymour and Cooper 1999, Pearl 2000).

The primary inference mechanism in stochastic domains is some form of Bayes' rule. As we noted in Section 5.3, however, the full use of Bayesian inference in complex domains quickly becomes intractable. Probabilistic graphical models are specifically designed to address this complexity. Probabilistic graphical models are "... a marriage between probability theory and graph theory" (Jordan 1999).

Probabilistic graphical models are also becoming more important in machine learning (Chapter 13) because they can address the problems of uncertainty and complexity, problems that are present, and can be fundamentally limiting, for modern AI. Jordan points out that the problem of modularity is the cornerstone of graphical modules: simple parts of a model are combined together using probability theory which supports the consistency of the system as a whole and at the same time integrates the graphical model with the data of an application (Jordan 1999). At the same time graph theory offers graphical models both

an intuitive way of representing highly interacting sets of components as well as a data structure that supports efficient inference algorithms.

In Section 9.3 and again in Chapter 13 we consider two types of probabilistic graphical models: the directed, Bayesian belief networks and various forms of Markov models, and undirected, clique trees and Markov random fields. Markov fields and other undirected graphical models are able to represent many cyclic dependencies that are not possible to represent with directed graphs. On the other hand, Bayesian belief networks are directed graphical models that are able to capture implicit inferred relationships and dependencies more precisely and efficiently than with undirected graphical models.

In the following sections we present Bayesian belief networks and several inference techniques specifically designed to address the computational complexity of undirected graphs and/or full Baysian reasoning. Later, in Section 9.3.5, we introduce Markov processes and mention several important representational variations. One of these is the dynamic Bayesian network and another the discrete Markov process.

One of the major limitations of BBNs and traditional HMMs is their inherent propositional nature. To address this limitation, several attempts have been made to merge prepositional logic, the full variable-based predicate calculus; with probabilistic graphical models. We summarize several of these efforts in Section 9.3.7, and visit first-order probabilistic systems again in Chapter 13.

### 9.3.1 A Directed Graphical Model: The Bayesian Belief Network

Although Bayesian probability theory, as presented in Chapter 5, offers a mathematical foundation for reasoning under uncertain conditions, the complexity encountered in applying it to realistic problem domains can be 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* or *BBNs* (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 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 are able 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. BBNs make these intuitions precise.

As an example Bayesian belief network, consider again the traffic problem of Section 5.4, represented by Figure 9.14. Recall that road construction was C, an accident, A, the presence of orange barrels, B, bad traffic, T, and flashing lights was L. To calculate the joint probability of all the parameters of the example, taking the full Bayesian approach, required knowledge or measurements for all parameters being in particular states. Thus,

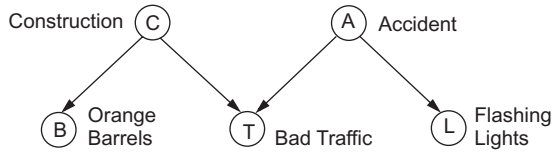


Figure 9.14 The graphical model for the traffic problem first introduced in Section 5.3.

the joint probability, using a topologically sorted order for variables, is:

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

The number of parameters in this joint probability is $2^5$ or 32. This table is exponential in the number of parameters involved. For a problem of any complexity, say with thirty or more parameters, the joint distribution table would have more than a billion elements!

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:

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

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, the probabilistic relationship 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. 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 billion 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 (Section 3.1.1), 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 the traffic example we have an even stronger situation, with no undirected cycles. This allows us to calculate very simply the probability distribution at every node. The distribution of nodes with 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 correlations between the parents of other non-descendent nodes. This produces a natural abductive separation where accident has no correlation at all with the presence of orange barrels, as is seen in Figure 9.14.

We summarize our discussion of BBNs with the following definition.

DEFINITION

BAYESIAN BELIEF NETWORK

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* (or *temporal*) *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 further in Section 9.3.4 and in Chapter 13; see also Friedman (1998) or Ghahramani and Jordan (1997).

In the next section we consider an assumption implicit in much of expert human reasoning: that the presence or absence of data in a (implicitly causal) domain can partition and focus the search for explanations within that domain. This fact also has important complexity implications for exploring a search space.

### 9.3.2 Directed Graphical Models: d-separation

An important advantage of representing application domains as graphical models is that the presence or absence of information can lead to partitioning the model and as a result controlling the complexity of the search. We next present several examples of this, and then offer the definition of *d-separation* that supports these intuitions.

Let us consider the diagnosis of oil problems in a car engine: suppose that worn piston rings cause excessive oil consumption which in turn causes a low oil level reading for the car. This situation is reflected by Figure 9.15a, where A is worn piston rings, V is excessive oil consumption, and B is low oil level. Now, if we do not know anything about excessive oil consumption, then we have a causal relationship between worn piston rings and low oil level. However, if some test gives knowledge of the state of excessive oil consumption, then worn piston rings and low oil level are independent of each other.

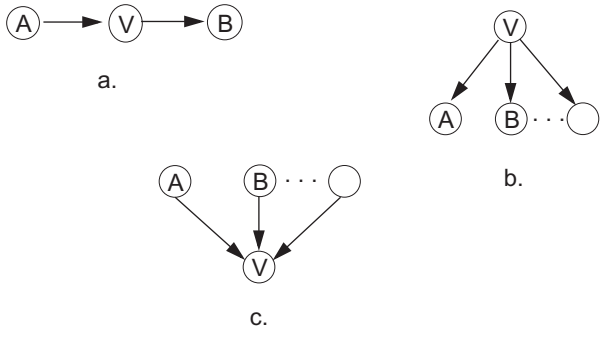


Figure 9.15 Figure 9.15a is a serial connection of nodes where influence runs between A and B unless V is instantiated. Figure 9.15b is a diverging connection, where influence runs between V s children, unless V is instantiated. In Figure 9.15c, a converging connection, if nothing is known about V then its parents are independent, otherwise correlations exist between its parents.

In a second example: suppose that worn piston rings can cause both blue exhaust as well as low oil level. This situation is shown in Figure 9.15b, with V as worn piston rings, A as blue exhaust, and B is low oil level. If we know that worn piston rings is either true or false then we don't know whether blue exhaust and low oil level are correlated; if we have no information on worn piston rings then blue exhaust and low oil level are correlated.

Finally, if low oil level can be caused by either excessive oil consumption or by an oil leak then, given knowledge of whether there is low oil level, its two possible causes are correlated. If the state of low oil level is unknown, then its two possible causes are independent. Furthermore, if low oil level is true then establishing oil leak as true will explain away excessive oil consumption. In either case, information on low oil level is a key element in the reasoning process. We see this situation in Figure 9.15c, with A as excessive oil consumption, B as oil leak, and V as low oil level.

We make these intuitions precise by defining the d-separation of nodes in a belief network or other graphical model (after Pearl 1988):

DEFINITION

d-SEPARATION

Two nodes A and B in a directed acyclic graph are *d-separated* if every path between them is blocked. A path is any continuous series of connections in the graph (linking nodes in any direction, e.g., there is a path from A to B in Figure 9.15b). A path is blocked if there is an intermediate node V in the path with either of the properties:

the connection is *serial* or *diverging* and the state of V is known, or

the connection is *converging* and neither V nor any of V's children have evidence.

We give further instances of serial, diverging, and converging node relationships, as well as how d-separation influences argument paths in the example of Figure 9.16.

Before leaving the graphs of Figure 9.15, we demonstrate how the assumptions of a Bayesian belief network can simplify the computation of conditional probabilities. Using Bayes law, any joint probability distribution can be decomposed into a product of conditional probabilities. Thus, in Figure 9.15a. the joint probability of the three variables A, V, B is:

p(A,V,B) = p(A) p(V|A) p(B|A,V).

We use the assumption of a Bayesian belief network, that the conditional probability of a variable given knowledge of all its predecessors is equal to its conditional probability given knowledge of only its parents. As a result, in the equation above p(B|A,V) becomes p(B|V) because V is a direct parent of B and A is not. The joint probability distributions for the three networks of Figure 9.15 are:

a) p(A,V,B) = p(A) p(V|A) p(B|V),
b) p(V,A,B) = p(V) p(A|V) p(B|V), and
c) p(A,B,V) = p(A) p(B) p(V|A,B).

As the traffic example showed (Figure 9.14), for larger Bayesian belief networks, many more variables in the conditional probabilities can be eliminated. It is this simplification that makes Bayesian belief networks and other graphical models far more statistically tractable than a full Bayesian analysis. We next present a more complex graphical model, one containing an undirected cycle, and propose an efficient inference algorithm, *clique tree propagation*.

### 9.3.3 Directed Graphical Models: An Inference Algorithm

The next example, adapted from Pearl (1988), shows a more complex Bayesian network. In Figure 9.16, 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 a cycle if the graph is undirected.

We 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) = p(W|S) p(S) or p(R) = p(R|S) p(S). The two causes of WS are not mutually independent; for example, if S = summer,

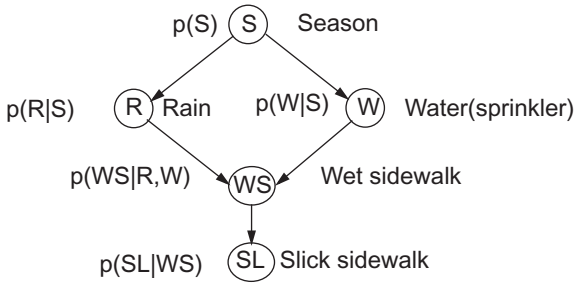


Figure 9.16 An example of a Bayesian probabilistic network, where the probability dependencies are located next to each node. This example is from Pearl (1988).

then p(W) and p(R) could both go up. Thus the complete correlations of the two variables, along with their further correlation with 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 Table 9.4. We now calculate one entry in that table, x, where R and W are both true; to make life simpler we assume the season S is either hot or cold.

x = p( R = t, W = t) for the two (hot, cold) 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 Table 9.4 can be completed. This makes up the joint probability for rain and water from the sprinkler. This larger "macro element" represents p(WS) = p(WS | R,W) p(R,W). We have gotten away with a rather reasonable calculation here; the problem is, as noted above, that this calculation is exponential in the number of parents of the state.

| R | W | p(WS) | |
|---|---|---|---|
| t | t | x | S=hot |
| t | f | | S=cold |
| f | t | | |
| f | f | | |

Table 9.4 The probability distribution for p(WS), a function of p(W) and p(R), given the effect of S. We calculate the effect for x, where 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 9.16 with an acyclic *clique tree*, as seen in Figure 9.17. The rectangular boxes of Figure 9.17a 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 9.17b, to pass information through the full tree of cliques, called the *junction tree*. We next present a algorithm developed by Laurintzen 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 9.17b).
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 9.17b.
4. Form the junction tree with the resulting triangulated structure. This is done 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 resulting *junction tree* is created by connecting any two junctions that share at least one variable, as in Figure 9.17a.

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.

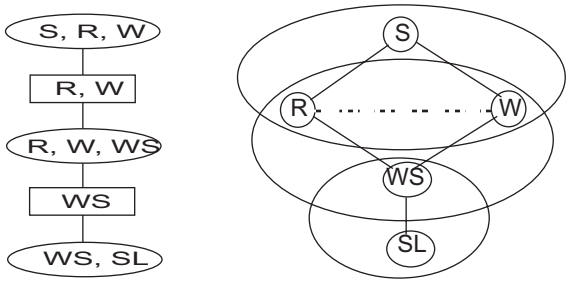


Figure 9.17 A junction tree (a) for the Bayesian probabilistic network of (b). Note that we started to construct the transition table for the rectangle R,W.

Unfortunately, this decision of deigning 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 9.17 are 2*2*2, 2*2*2, and 2*2.

Finally, we take the example network of Figure 9.16 and return to the issue of d-separation. Remember, the point of d-separation is that with some information, parts of the belief network can be ignored in calculating probability distributions.

1. SL is d-separated from R, S, and W, given that WS is known.
2. d-separation is symmetric, that is, S is also d-separated from (and not a possible explanation of SL), given knowledge of WS.
3. R and W are dependent because of S, but knowing S, R and W are d-separated.
4. If we know WS, then R and W are not d-separated; if we don't know S, then R and W are.
5. Given the chain R -> WS -> SL, if we know WS, then R and SL are d-separated.

We must be careful when we know information about the descendents of a particular state. For example, if we know SL, then R and W are NOT d-separated, since SL is correlated with WS, and knowing WS, R and W are not d-separated.

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.

There are a number of algorithms available 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.

In the next section we introduce dynamic Bayesian networks.

### 9.3.4 Directed Graphical Models: Dynamic Bayesian Networks

We next consider a generalization of Bayesian belief networks, the *dynamic* (or *temporal*) *Bayesian network*, or *DBN*. The idea supporting DBNs is to represent a state of the domain or an observation time with a set of random variables, not just with a single random variable. With this extension Bayesian belief networks can be used to represent the conditional independence between variables from different perspectives, for example, across time periods.

Most of the events that we humans encounter, and to which we are expected to react

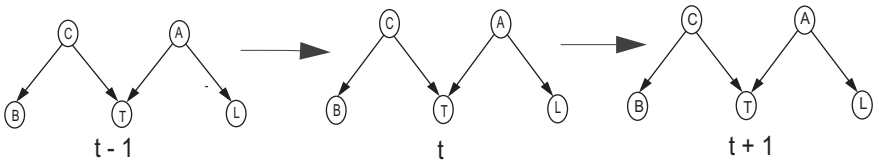


Figure 9.18 The traffic problem, Figure 9.14, represented as a dynamic Bayesian network.

intelligently, are temporal: they present themselves across periods of time. Two examples can be seen in Figures 9.18 and 9.19. In Figure 9.18 we have taken our traffic model of Figure 9.14 and mapped it across time periods. This might capture a driver's changing viewpoint across time, every minute say, is captured by the changing parameters of a Bayesian network. At the first time period the driver is just aware of slowing down, at the second time period she notices the presence of orange barrels, at the third time period, there are even more barrels, etc. The DBN is expected to capture the growing awareness of road construction, along with a (much) lesser confidence of an accident.

Figure 9.19 is sample data of signals coming from a complex environment. (These are actually multiple heat, vibration, and other signals coming from the rotor system of a helicopter). What the DBN in this situation must measure is how these signals change over time and advise the controller of the current "running health" of the system. We consider this example further, as well as define the DBN more precisely, in Section 13.2.

DBNs are most often employed to track two types of time varying systems. First, time dependent sequences such as described in Figures 9.18 and 9.19, and second phenomena naturally occurring across time such as phoneme or word sampling in a natural language analysis application. Note that it is possible to represent correlations between random variables in the same time slice (instantaneous correlations) with both the directed and undirected edges of graphical models. However, correlations between elements of graphs reflecting time series data must be captured with directed graphical models.

To summarize: If every edge connecting a series of graphical models representing time-related data is directed to reflect the time dimension, the model is called a dynamic Bayesian network. The DBN can also capture non-temporal sequential data such as language where the new information reflects change of state of processing. For further detail on DBNs see Friedman (1998) or Ghahramani and Jordan (1997) and Section 13.2.

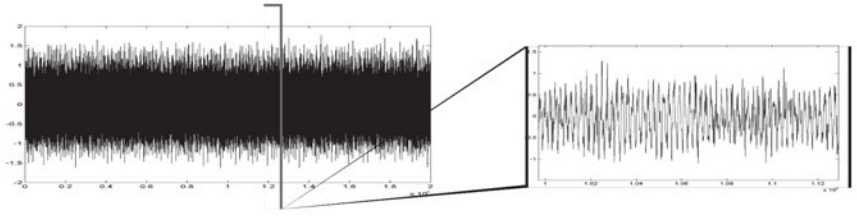

Figure 9.19 Typical time series data to be analyzed by a dynamic Bayesian network.

### 9.3.5 Markov Models: The Discrete Markov Process

In Section 3.1.2 we presented *finite state machines* as graphical representations where states are transitioned depending on the content of an input stream. The states and their transitions reflect properties of a formal language. We then presented a *finite state acceptor* (the Moore machine), a state machine that was able to "recognize" strings having various properties. In Section 5.3 we presented the *probabilistic finite state machine,* a state machine where the next state function was represented by a probability distribution on the current state. The *discrete Markov process* is a specialization of this approach, where the system ignores its input values.

Figure 9.20 is a *Markov state machine*, sometimes called a *Markov chain*, with four distinct states. This general class of system may be described at any time period as being in one of a set of S 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 $t_1$, $t_2$, $t_3$, ... , $t_t$. 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 $\sigma_t$.

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

$$p(\sigma_t) = p(\sigma_t \mid \sigma_{t-1}, \sigma_{t-2}, \sigma_{t-3}, ...)$$

where the $\sigma_{t-i}$ are the predecessor states of $\sigma_t$. In a *first-order Markov chain*, the probability of the present state is a function *only* of its direct predecessor state:

$$p(\sigma_t) = p(\sigma_t \mid \sigma_{t-1})$$

where $\sigma_{t-1}$ precedes $\sigma_t$. (In general, a state in a *Markov chain of order n*, is dependant on its preceding n states.) We also assume the right side of this equation is time invariant, i.e., we hypothesize that across all time periods of the system, the transitions between

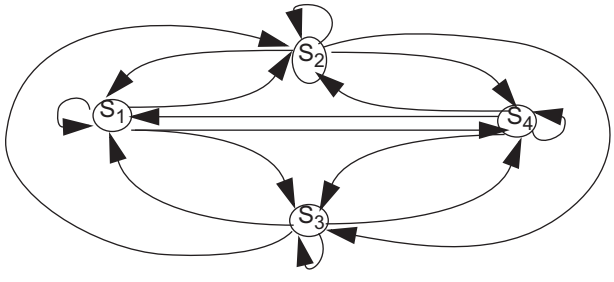


Figure 9.18 A Markov state machine or Markov chain with four states, $s_1$,...,$s_4$.

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 $s_j$:

$$a_{ij} = p(\sigma_t = s_i \mid \sigma_{t-1} = s_j), \quad 1 \leq i,j \leq N$$

Recall 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 $s_i$:

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

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 and then give an example.

D E F I N I T I O N

(OBSERVABLE) MARKOV MODEL

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.

Note that any probability distribution has the property of being a Markov model. The power of this approach comes from the first-order assumptions. 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. The time intervals for the Markov model will be noon each consecutive day. We also assume the transitions 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 9.20, and is supported by the matrix of state transitions $a_{ij}$:

In this $a_{ij}$ transition matrix, similar to the finite state machine in Section 3.1.2, the first row represents the transitions from $s_1$ to each of the states, including staying in the same state; the second row is the transitions from $s_2$, and so on. The properties required for the transitions to be probability distributions from each state are met (they sum to 1.0).

$a_{ij}$ =

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ |
|---|---|---|---|---|
| $s_1$ | 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 |

We now ask questions of our model. Suppose today, $s_1$, is sunny; what is the probability of the next five days remaining sunny? Or what is the probability of the next five days being sunny, sunny, cloudy, cloudy, precipitation? To solve this second problem, we determine the probability and that 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 these 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 follows from the first-order Markov assumption where weather each day is a function (only) of the weather the day before. We also observe 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 $s_i \neq s_j$, 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:

$$d_i = \sum_{d=1}^{n} d(a_{ii})^{(d-1)}(1 - a_{ii}) \quad \text{where } n \text{ approaches } \infty, \text{ or:}$$

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

For example, the expected number of consecutive precipitation days, given this model, is 1/(1 - .3) or 1.43. Similarly, the number of consecutive sunny days one might expect with this model is 1.67. We next summarize various forms of Markov models, many of which we see in more detail in Chapter 13.

### 9.3.6 Markov Models: Variations

In the Markov models we have seen to this point, each state corresponds to a discrete physical - and observable - event, such as the value of weather at a certain time of day. This class of model is really fairly limited and we now generalize it to a wider class of models. We describe several of these approaches briefly. Section 13.1 offers a more comprehensive presentation of hidden Markov models, along with several important variations.

**Hidden Markov Models**

In the Markov models already presented, each state corresponds to a discrete physical and observable event. In a *hidden Markov model*, or *HMM*, the observed values are assumed to be probabilistic functions of a current hidden state.

For example, an observed human's speech sound is a reflection of some phone the speaker is intending to produce. The uttered sound is only probabilistically related to the actual state or intent of the speaker. Thus, the HMM is a doubly embedded stochastic process where the observable stochastic process (the speaker's noise) is supported by an unobservable stochastic process (the speaker's state or intent). The HMM is a special case of the DBNs as we see in Section 9.3.2. For more detail on HMMs see Section 13.1 and 13.2 and Rabiner (1989).

**Semi-Markov Models**

A *semi-Markov model* is a two-component Markov chain. The first component is responsible for the choice of the *next state transition* and the second component is responsible for the *transition time*. When a semi-Markov model enters a state $s_i$ it remains in that state for time $t_i$, which is taken from a state duration probability distribution. Once time $t_i$ is passed, the process transitions to the next state $s_{i+1}$ according to the state transition probability distribution, and a state transition time, $t_{i+1}$ is selected again. A semi-Markov model supports arbitrary state duration probability distributions as compared to the fixed time transitions specified by traditional Markov models.

**Markov Decision Processes**

Two other variants of Markov models, used extensively in reinforcement learning, are the *Markov decision process*, or *MDP*, and the *partially observable markov decision process*, or *POMDP*. The MDP is defined on two spaces: the state space for the problem under consideration and the space of possible actions. The transition to a new state in the state space

is dependent on the current state and current action and is guided by a conditional probability distribution. At each state a reward is computed given the current state and an action. Typically, the task of reinforcement learning is to maximize a cumulative reward function under current conditions.

While the MDP works with an observed (deterministic) state before using a probability based transition matrix to select its next state, the POMDP has only probabilistic knowledge of its current state (as well as the probabilistic matrix for determining its next state). Thus the POMDP can prove computationally intractable for complex environments. We consider MDPs and POMDPs further and reinforcement learning in Sections 10.7 and 13.3.

### 9.3.7 First-Order Alternatives to BBNs for Probabilistic Modeling

To this point in Section 9.3 we have presented propositional calculus based representations for reasoning with uncertainty. It is only natural to ask in what sense *probabilistic predicate logic* based models might be used for reasoning. The strength of the full predicate logic representation is a declarative semantics along with a variable-based representational framework.The representational limitations of the traditional propositional logic based approaches include a limited ability to cope with noise and uncertainty as well as a static view of representations.

Thus the Baysian and Markovian models presented in Section 9.3 are limited to a propositional level representation, where it can be cumbersome to represent general (variable based) relationships with a distribution, for example, $\forall X$ male(X) $\rightarrow$ smart(X) (0.49 0.51). Many complex problem domains require this level of expressiveness.

It is also important for complex domains, especially in the areas of diagnosis and prognosis, to be able to represent repetitive, recursive, and (potentially) infinite structures. Therefore, if we want to represent systems that change state over time we must have that representational power. This requires a recursive, or repeat-until-done type control regime. If we wish to dynamically rebuild a traditional BBN, we are forced to reconstruct the entire network over time periods because these structures lack a declarative and time-oriented semantics.

Recent research activity has created a variety of important extensions to graphical models, including a number of first-order (variable-based and recursive) systems. There are also a number of hierarchical and decomposable Bayesian models. We list several of these new modeling formalisms that support such a representation. We give examples of several of these representations when we present probabilistic approaches incorporated with machine learning in Chapter 13.

**Bayesian Network Construction from Knowledge Bases** (Haddawy 1994, Ngo and Haddawy 1997, Ngo et al. 1997).

Haddawy, Ngo and colleagues merge together a first-order logic system with Bayesian belief networks. As a result, they create a first-order probabilistic logic used as a representational language to specify a certain class of networks in the form of a knowledge base.

The formal semantics of the (Bayesian) knowledge base is enforced by using a strictly defined representational language specifying the sentences of the base. Haddawy and Ngo provided a provably correct Bayesian network generation algorithm implementing a combination of a formal semantics for the underlying logical language as well as an independence semantics for the knowledge base. One feature of the generating algorithm is that it avoids producing irrelevant nodes for the network by using user specified evidence.

Haddawy and Ngo argue that reasoning becomes very inefficient when a knowledge base becomes very large. As a remedy they propose to use context information to index the probabilistic sentences of a knowledge base. When a model generation algorithm constructs a network, it omits the sentences from the knowledge base whose context is not relevant to a current task. The resulting model is significantly smaller than if the entire knowledge base is used for model construction. This approach by Haddawy, Ngo, and colleagues is one of the first research attempts in the field of stochastic logic modeling that explicitly uses contextual information about a domain as a way to focus the knowledge base on the relevant information and reduce the size of a model necessary to solve a particular reasoning task.

**Bayesian Logic Programs, BLPs** (Kersting and DeRaedt 2000).

*Bayesian logic programs* offer a representational framework including logic programming and graphical models. More specifically, BLPs combine Bayesian belief networks with Horn clause logic (Prolog, see Section 14.3). Given a specific query the system generates a Bayesian network to answer the query using a set of first-order rules with uncertainty parameters. The resulting representation is easy to interpret and its designers assign it a model theoretic semantics similar to that seen in Section 2.3.

**Probabilistic Relational Models**, **PRM**s (Friedman et al. 1999, Getoor et al. 2001)

Friedman, Getoor, and their colleagues have developed another approach to first-order probabilistic systems. Rather than creating a declarative logic-like language, PRMs specify a probability model on classes of objects. PRMs define probability constraints at the class level so that these constraints can be used with any object from that class. With PRMs it is also possible to specify probabilistic dependencies between attributes of related classes. A number of extensions to PRMs allow sub-classing that supports probabilistic dependencies at appropriate levels of detail (Getoor et al. 2001). Class hierarchies allow modeling both at the level of individuals (the traditional Bayesian belief network) as well as for classes. A further extension of PRMs is to treat the relational structures of models themselves as uncertain rather than fixed. From this perspective there is a close relationship between PRMs and relational databases, as they can have similar structures.

**Markov Logic Networks, MLNs** (Richardson and Domingos 2006).

Richardson and Domingos propose another probabilistic logic-based system called Markov logic networks (MLNs). Most of the previous logic-based approaches to stochastic modeling use restricted subsets of general logic, with Horn clauses being the most

usual representation. In order to remove restrictions on logic, Richardson and Domingos use the general first-order logic, whose sentences are converted into a conjunctive normal form (CNF). They also use Markov random fields (Pearl 1988) as a probabilistic counterpart of logic, another major difference from the previously described systems (Bayesian belief networks are by far the most used representation). Consequently, the mapping from logical sentences in CNF to Markov random fields is straightforward. MLNs are the first theoretical framework with a complete mapping from the first-order predicate calculus with function symbols to probability distributions.

**Loopy Logic** (Pless et al. 2006, Chakrabarti et. al. 2006, Sakhanenko et al. 2006)

Loopy logic is a first-order and Turing-complete declarative probabilistic language. It gets its name from its use of the *loopy belief propagation* algorithm, Pearl (1998), for inference. Its logic-based declarative representations are first translated into Markov random fields. Parameter learning with the expectation maximization (EM) algorithm meshes well with loopy belief propagation. We demonstrate the loopy logic system, together with its use of the Markov random field and EM-based parameter learning in Section 13.2.

There are a number of further extensions to traditional BBNs, that make them first-order and Turing complete. Further examples of first-order stochastic modeling include the *IBAL* language of Pffefer (2001), and the *Stochastic Lambda Calculus* of Pless and Luger (2002). Object-oriented stochastic representations include (Koller and Pfeffer 1997, 1998; Pfeffer et al 1999; Xiang et al. 2000; Pless et al. 2000). Stochastic methods are important across the field of AI; see, for example, probabilistic agents (Kosoresow 1993). We see stochastic methods applied to machine learning in Chapter 13 and to natural language processing in Section 15.4.

## 9.4 Epilogue and References

From the beginning of the AI research enterprise, there has been an important subset of the community that feels logic and its extensions offer a sufficient representation for the characterization of intelligence. Important alternatives to first-order predicate calculus have been proposed for describing reasoning under conditions of uncertainty:

1. **Multiple-valued logics.** These extend logic by adding new truth values such as unknown to the standard values of true and false. This can, for example, provide a vehicle for distinguishing between assertions that are known to be false and those that are simply not known to be true.
2. **Modal logics.** Modal logic adds operators that enable a logic system to deal with problems of knowledge and belief, necessity and possibility. We discussed modal operators for *unless* and *is consistent with* in the present chapter.
3. **Temporal logics.** Temporal logics enable logic to quantify expressions with regard to time, indicating, for example, that an expression is *always* true or *will be true at some time in the future*.

4. **Higher-order logics.** Many categories of knowledge involve higher-order concepts, where predicates, and not just variables, may be quantified. Do we really need higher-order logics to deal with this knowledge, or can it all be done in first-order logic? If higher-order logics are needed, how may they best be formulated?
5. **Logical formulations of definitions, prototypes, and exceptions.** Exceptions are often seen as a necessary feature of a definitional system. However, careless use of exceptions undermines the semantics of a representation. Another issue is the difference between a definition and a prototype, or representation of a *typical* individual. What is the exact difference between the properties of a class and the properties of a typical member? How should prototypical individuals be represented? When is a prototype more appropriate than a definition?

Logic-based representations continue to be an important area for research (McCarthy 1968, Hayes 1979, Weyhrauch 1980, Moore 1982, Turner 1984).

There are several other important contributions to truth maintenance system (TMS) reasoning. *Logic-based TMS* is based on the work of McAllester (1978). In LTMS relationships between propositions are represented by clauses which can be used to deduce the truth values of any of the propositions they describe. Another approach, the *multiple belief reasoner* MBR is like the ATMS reasoner (deKleer 1984); similar ideas may be found in Martins and Shapiro (1983). MBR is based on a logic language called SWM* which describes knowledge states. Algorithms for inconsistency checking across reasoning in the knowledge base may be found in Ginsburg (1997) and Martins (1990, 1991). For further information on the node algebra support for JTMS see Doyle (1983) or Reinfrank (1989). Default logic allows any theorem inferred in an extension of a system to be admitted as an axiom for further reasoning. Reiter and Criscuolo (1981) and Touretzky (1986) develop these issues.

There is a rich literature on nonmonotonic reasoning, belief logics, and truth maintenance, besides the original papers in the area (Doyle 1979; Reiter 1985; deKleer 1986; McCarthy 1977, 1980). For stochastic models see *Probabilistic Reasoning in Intelligent Systems* by Pearl (1988), *Readings in Uncertain Reasoning* by Shafer and Pearl (1990), *Representations of Commonsense Knowledge* by Davis (1990), and numerous articles in recent AAAI, UAI, and IJCAI proceedings. We recommend *The Encyclopedia of Artificial Intelligence*, by Stuart Shapiro (2nd edition, 1992), for coverage of many of the reasoning models of this chapter. Josephson and Josephson (1994) have edited a collection of papers in *Abductive Inference: Computation, Philosophy, and Technology.* Also see *Formal Theories of the Commonsense World* (Hobbs and Moore 1985). In *Causality*, Pearl (2000) makes a contribution to understanding the notion of cause-effect relations in the world.

Further research in circumscriptive and minimum model logics may be found in Genesereth and Nilsson (1987), Lifschitz (1986), and McCarthy (1986). Another contribution to circumscriptive inference is Perlis' (1988) reasoning about a particular agent's lack of knowledge. Ginsburg (1987), has edited a collection of papers on nonmonotonic systems, *Readings in Nonmonotonic Reasoning*.

For further reading on fuzzy systems we recommend the original paper by Lotfi Zadeh (1983) and the more modern integrations of this technology found in *Fuzzy Sets, Neural Networks and Soft Computing* by Yager and Zadeh (1994), and *Fuzzy Logic with*

*Engineering Applications* by Timothy Ross (1995). The solution to the inverted pendulum problem presented in Section 9.2.2 was adapted from Ross.

Algorithms for Bayesian belief network inference include Pearl's (1988) *message passing* and *clique tree triangulation* (Lauritzen and Spiegelhalter 1988, see Section 9.3). For further discussion of these algorithms see the *Encyclopedia of AI* (Shapiro 1992) and (Huang and Darwiche 1996). The spring 1996 issue of the *AISB Quarterly* contains an introduction to *Bayesian belief networks* (van der Gaag 1996); we also recommend the discussions of *qualitative probabilistic networks* by Wellman (1990) and Druzdel (1996).

Stochastic representations and algorithms continue to be a very active research area (Xiang et al. 1993, Laskey and Mahoney 1997). Limitations of Bayesian representations have motivated research in hierarchical and composable Bayesian models (Koller and Pfeffer 1997, 1998, Pfeffer et al. 1999, Xiang et al. 2000). Further extensions of the propositional based graphic models we have presented can be found in the literature. We recommend reading Koller and Pfeffer (1998) and Pless et al. (2000) for ideas on object-oriented stochastic representations. The *IBAL* language of Pfeffer (2001) and the *Stochastic Lambda Calculus* of Pless and Luger (2002) are examples of first-order stochastic functional languages. Cussens (2001), and Pless and Luger (2003) have created first-order logic-based (declarative) languages for stochastic inference. See also the many references given in Section 9.3.7, which have not been repeated here.

We see many issues from Section 9.3 again, especially in Part IV, Chapter 13, the presentation of probabilistic approaches to machine learning.

## 9.5 Exercises

1. Identify three application domains where reasoning under conditions of uncertainty is necessary. Pick one of these areas and design six inference rules reflecting reasoning in that domain.
2. Given the following rules in a "back-chaining" expert system application:

   $A \wedge \text{not}(B) \Rightarrow C$ (.9)

   $C \vee D \Rightarrow E$ (.75)

   $F \Rightarrow A$ (.6)

   $G \Rightarrow D$ (.8)

   The system can conclude the following facts (with confidences):

   F(.9)

   B(−.8)

   G(.7)

   Use the Stanford certainty factor algebra to determine E and its confidence.
3. Consider the simple MYCIN-like rule: if $A \wedge (B \vee C) \Rightarrow D$ (.9) $\wedge$ E (.75). Discuss the issues that arise in capturing these uncertainties in a Bayesian context. How might this rule be handled in Dempster–Shafer reasoning?

4. Create a new example of diagnostic reasoning and use the Dempster–Shafer equations of Section 9.2.3 to obtain belief distributions as in Table 9.1 and 9.2.

5. Use the schema axioms presented in McCarthy (1980, Section 4) to create the circumscription results presented in Section 9.1.3.

6. Create another reasoning network similar to that of Figure 9.4 and show the dependency lattice for its premises, as was done in Figure 9.5.

7. Reasoning by assumption of a minimum model is important in human everyday life. Work out two more examples that assume minimum models.

8. Continue the inverted pendulum example of Section 9.2.2 with two more iterations of the controller where the output of one iteration provides the input values for the next iteration.

9. Write a program that implements the fuzzy controller of Section 9.2.2.

10. Go to the literature, for example Ross (1995), and describe two other areas where fuzzy control might be appropriate. Construct a set of fuzzy rules for those domains.

11. Tom has invested some of his savings in a portfolio of five stocks, H = {Bank of China, Citibank, Intel, Nokia, Legend}. These stocks can be classified into *Bank stocks* (Bank of China and Citibank), *Hi-tech stocks* (Intel, Nokia, and Legend) and *China stocks* (Bank of China and Legend). You have offered to build an expert system that can give him advice as to how he should manage his stock portfolio. To do so, you interviewed many financial consultants and came up with a set of rules for the expert system. These rules tell Tom how he should divide up his money among the different stocks in his portfolio. The proportions recommended are directly proportional to the probability of the increase of the price of the stocks or the type of stock concerned:

    R1: IF the interest rate goes up THEN buy Bank Stocks, Hi-tech stocks and Nokia in the proportion of 0.8, 0.15, and 0.05 respectively.

    R2: IF employment rate goes up THEN buy Bank of China, Intel and the Hi-Tech stocks in the proportion of 0.5, 0.2, and 0.3 respectively.

    R3: IF the inflation rate comes down THEN buy Intel, Citibank and Hi-Tech stocks in the proportion of 0.1, 0.4 and 0.5 respectively

    David noticed that the interest rate has just gone up, using Dempster-Shafer's theory of evidence, calculate the belief intervals for the three types of stock, i.e., the Bank stock, the China stock, and the Hi-Tech stock, and for Nokia. Is there any difference in the belief and the belief intervals between Nokia and the Hi-Tech stocks? Why or why not? (This problem was proposed by James Liu Nga Kwok at Hong Kong Polytechnic University, The solution in the Instructor Guide is by a student, Qi Xianming).

12. Put another link in Figure 9.16, say connecting season directly to slick sidewalk and then create a clique tree to represent this situation. Compare the complexity issues with those of the clique tree of Figure 9.17.

13. Complete the symbolic evaluations that are required to finish Table 9.4.

14. Create an algorithm for Bayesian belief propagation and apply it to the slippery sidewalk domain of Section 9.3.2. You might use Pearl's (1988) message passing approach or the clique triangulation method proposed by Lauritzen and Spiegelhalter (1988).

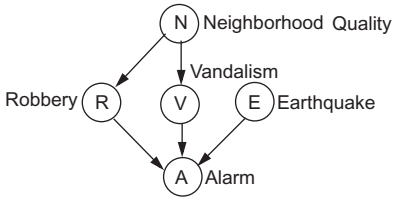


Figure 9.23 A belief network representing the possibility of a house alarm in a dangerous neighborhood.

15. Create a Bayesian belief diagram for another application, for example medical diagnosis, geological discovery, or automobile fault analysis. Point out examples of d-separation and create a clique tree for this network.
16. Create cliques and a junction tree for the following situation (seen in Figure 9.23). Robbery, vandalism and an earthquake can all set off (cause) a house alarm. There is also a measure of the potential dangers in the neighborhood of the house.
17. Take the diagnostic reasoning situation developed in Table 9.1 and 9.2 of the Dempster–Shafer model of Section 9.2.3 and recast it as a Bayesian Belief network. Compare and contrast these two approaches to diagnosis.
18. Given that you wanted to design a second-order Markov model, i.e., where each observable state would be dependent on the previous two observable states. How would you do this? What would the transition probability matrix look like?
19. Given the observable Markov model of weather of Section 9.3.4:
    a. Determine the probability that (exactly) the next five days will be sun.
    b. What is the probability of exactly three days of sun, then one day of precipitation, and then exactly one day of sun?