# MODULE-5

## 13.1 Acting under Uncertainty

UNCERTAINTY

Agents may need to handle **uncertainty**, whether due to partial observability, nondeterminism, or a combination of the two. An agent may never know for certain what state it's in or where it will end up after a sequence of actions.

We have seen problem-solving agents (Chapter 4) and logical agents (Chapters 7 and 11) designed to handle uncertainty by keeping track of a **belief state**—a representation of the set of all possible world states that it might be in—and generating a contingency plan that handles every possible eventuality that its sensors may report during execution. Despite its many virtues, however, this approach has significant drawbacks when taken literally as a recipe for creating agent programs:

- When interpreting partial sensor information, a logical agent must consider *every logically possible* explanation for the observations, no matter how unlikely. This leads to impossible large and complex belief-state representations.
- A correct contingent plan that handles every eventuality can grow arbitrarily large and must consider arbitrarily unlikely contingencies.
- Sometimes there is no plan that is guaranteed to achieve the goal—yet the agent must act. It must have some way to compare the merits of plans that are not guaranteed.

Suppose, for example, that an automated taxi!automated has the goal of delivering a passenger to the airport on time. The agent forms a plan, $A_{90}$, that involves leaving home 90 minutes before the flight departs and driving at a reasonable speed. Even though the airport is only about 5 miles away, a logical taxi agent will not be able to conclude with certainty that "Plan $A_{90}$ will get us to the airport in time." Instead, it reaches the weaker conclusion "Plan $A_{90}$ will get us to the airport in time, as long as the car doesn't break down or run out of gas, and I don't get into an accident, and there are no accidents on the bridge, and the plane doesn't leave early, and no meteorite hits the car, and . . . ." None of these conditions can be deduced for sure, so the plan's success cannot be inferred. This is the **qualification problem** (page 268), for which we so far have seen no real solution.

Nonetheless, in some sense $A_{90}$ *is* in fact the right thing to do. What do we mean by this? As we discussed in Chapter 2, we mean that out of all the plans that could be executed, $A_{90}$ is expected to maximize the agent's performance measure (where the expectation is relative to the agent's knowledge about the environment). The performance measure includes getting to the airport in time for the flight, avoiding a long, unproductive wait at the airport, and avoiding speeding tickets along the way. The agent's knowledge cannot guarantee any of these outcomes for $A_{90}$, but it can provide some degree of belief that they will be achieved. Other plans, such as $A_{180}$, might increase the agent's belief that it will get to the airport on time, but also increase the likelihood of a long wait. *The right thing to do—the **rational decision**—therefore depends on both the relative importance of various goals and the likelihood that, and degree to which, they will be achieved.* The remainder of this section hones these ideas, in preparation for the development of the general theories of uncertain reasoning and rational decisions that we present in this and subsequent chapters.

### 13.1.1 Summarizing uncertainty

Let's consider an example of uncertain reasoning: diagnosing a dental patient's toothache. Diagnosis—whether for medicine, automobile repair, or whatever—almost always involves uncertainty. Let us try to write rules for dental diagnosis using propositional logic, so that we can see how the logical approach breaks down. Consider the following simple rule:

$$Toothache \Rightarrow Cavity\ .$$

The problem is that this rule is wrong. Not all patients with toothaches have cavities; some of them have gum disease, an abscess, or one of several other problems:

$$Toothache \Rightarrow Cavity \lor GumProblem \lor Abscess \ldots$$

Unfortunately, in order to make the rule true, we have to add an almost unlimited list of possible problems. We could try turning the rule into a causal rule:

$$Cavity \Rightarrow Toothache\ .$$

But this rule is not right either; not all cavities cause pain. The only way to fix the rule is to make it logically exhaustive: to augment the left-hand side with all the qualifications required for a cavity to cause a toothache. Trying to use logic to cope with a domain like medical diagnosis thus fails for three main reasons:

LAZINESS
- **Laziness**: It is too much work to list the complete set of antecedents or consequents needed to ensure an exceptionless rule and too hard to use such rules.

THEORETICAL IGNORANCE
- **Theoretical ignorance**: Medical science has no complete theory for the domain.

PRACTICAL IGNORANCE
- **Practical ignorance**: Even if we know all the rules, we might be uncertain about a particular patient because not all the necessary tests have been or can be run.

The connection between toothaches and cavities is just not a logical consequence in either direction. This is typical of the medical domain, as well as most other judgmental domains: law, business, design, automobile repair, gardening, dating, and so on. The agent's knowledge

DEGREE OF BELIEF
PROBABILITY THEORY

can at best provide only a **degree of belief** in the relevant sentences. Our main tool for dealing with degrees of belief is **probability theory**. In the terminology of Section 8.1, the **ontological commitments** of logic and probability theory are the same—that the world is composed of facts that do or do not hold in any particular case—but the **epistemological commitments** are different: a logical agent believes each sentence to be true or false or has no opinion, whereas a probabilistic agent may have a numerical degree of belief between 0 (for sentences that are certainly false) and 1 (certainly true).

*Probability provides a way of* **summarizing** *the uncertainty that comes from our laziness and ignorance,* thereby solving the qualification problem. We might not know for sure what afflicts a particular patient, but we believe that there is, say, an 80% chance—that is, a probability of 0.8—that the patient who has a toothache has a cavity. That is, we expect that out of all the situations that are indistinguishable from the current situation as far as our knowledge goes, the patient will have a cavity in 80% of them. This belief could be derived from statistical data—80% of the toothache patients seen so far have had cavities—or from some general dental knowledge, or from a combination of evidence sources.

One confusing point is that at the time of our diagnosis, there is no uncertainty in the actual world: the patient either has a cavity or doesn't. So what does it mean to say the probability of a cavity is 0.8? Shouldn't it be either 0 or 1? The answer is that probability statements are made with respect to a knowledge state, not with respect to the real world. We say "The probability that the patient has a cavity, *given that she has a toothache*, is 0.8." If we later learn that the patient has a history of gum disease, we can make a different statement: "The probability that the patient has a cavity, given that she has a toothache and a history of gum disease, is 0.4." If we gather further conclusive evidence against a cavity, we can say "The probability that the patient has a cavity, given all we now know, is almost 0." Note that these statements do not contradict each other; each is a separate assertion about a different knowledge state.

### 13.1.2 Uncertainty and rational decisions

Consider again the $A_{90}$ plan for getting to the airport. Suppose it gives us a 97% chance of catching our flight. Does this mean it is a rational choice? Not necessarily: there might be other plans, such as $A_{180}$, with higher probabilities. If it is vital not to miss the flight, then it is worth risking the longer wait at the airport. What about $A_{1440}$, a plan that involves leaving home 24 hours in advance? In most circumstances, this is not a good choice, because although it almost guarantees getting there on time, it involves an intolerable wait—not to mention a possibly unpleasant diet of airport food.

PREFERENCE
OUTCOME
UTILITY THEORY

To make such choices, an agent must first have **preferences** between the different possible **outcomes** of the various plans. An outcome is a completely specified state, including such factors as whether the agent arrives on time and the length of the wait at the airport. We use **utility theory** to represent and reason with preferences. (The term **utility** is used here in the sense of "the quality of being useful," not in the sense of the electric company or water works.) Utility theory says that every state has a degree of usefulness, or utility, to an agent and that the agent will prefer states with higher utility.

The utility of a state is relative to an agent. For example, the utility of a state in which White has checkmated Black in a game of chess is obviously high for the agent playing White, but low for the agent playing Black. But we can't go strictly by the scores of 1, 1/2, and 0 that are dictated by the rules of tournament chess—some players (including the authors) might be thrilled with a draw against the world champion, whereas other players (including the former world champion) might not. There is no accounting for taste or preferences: you might think that an agent who prefers jalapeño bubble-gum ice cream to chocolate chocolate chip is odd or even misguided, but you could not say the agent is irrational. A utility function can account for any set of preferences—quirky or typical, noble or perverse. Note that utilities can account for altruism, simply by including the welfare of others as one of the factors.

Preferences, as expressed by utilities, are combined with probabilities in the general theory of rational decisions called **decision theory**:

DECISION THEORY

$$\textit{Decision theory} = \textit{probability theory} + \textit{utility theory}\ .$$

The fundamental idea of decision theory is that *an agent is rational if and only if it chooses the action that yields the highest expected utility, averaged over all the possible outcomes of the action.* This is called the principle of **maximum expected utility** (MEU). Note that "expected" might seem like a vague, hypothetical term, but as it is used here it has a precise meaning: it means the "average," or "statistical mean" of the outcomes, weighted by the probability of the outcome. We saw this principle in action in Chapter 5 when we touched briefly on optimal decisions in backgammon; it is in fact a completely general principle.

MAXIMUM EXPECTED UTILITY

Figure 13.1 sketches the structure of an agent that uses decision theory to select actions. The agent is identical, at an abstract level, to the agents described in Chapters 4 and 7 that maintain a belief state reflecting the history of percepts to date. The primary difference is that the decision-theoretic agent's belief state represents not just the *possibilities* for world states but also their *probabilities*. Given the belief state, the agent can make probabilistic predictions of action outcomes and hence select the action with highest expected utility. This chapter and the next concentrate on the task of representing and computing with probabilistic information in general. Chapter 15 deals with methods for the specific tasks of representing and updating the belief state over time and predicting the environment. Chapter 16 covers utility theory in more depth, and Chapter 17 develops algorithms for planning sequences of actions in uncertain environments.

## 13.2 Basic Probability Notation

For our agent to represent and use probabilistic information, we need a formal language. The language of probability theory has traditionally been informal, written by human mathematicians to other human mathematicians. Appendix A includes a standard introduction to elementary probability theory; here, we take an approach more suited to the needs of AI and more consistent with the concepts of formal logic.

```
function DT-AGENT(percept) returns an action
   persistent: belief_state, probabilistic beliefs about the current state of the world
               action, the agent's action

   update belief_state based on action and percept
   calculate outcome probabilities for actions,
      given action descriptions and current belief_state
   select action with highest expected utility
      given probabilities of outcomes and utility information
   return action
```

**Figure 13.1** A decision-theoretic agent that selects rational actions.

## 13.2.1 What probabilities are about

Like logical assertions, probabilistic assertions are about possible worlds. Whereas logical assertions say which possible worlds are strictly ruled out (all those in which the assertion is false), probabilistic assertions talk about how probable the various worlds are. In probability theory, the set of all possible worlds is called the **sample space**. The possible worlds are *mutually exclusive* and *exhaustive*—two possible worlds cannot both be the case, and one possible world must be the case. For example, if we are about to roll two (distinguishable) dice, there are 36 possible worlds to consider: (1,1), (1,2), $\ldots$, (6,6). The Greek letter $\Omega$ (uppercase omega) is used to refer to the sample space, and $\omega$ (lowercase omega) refers to elements of the space, that is, particular possible worlds.

SAMPLE SPACE

PROBABILITY MODEL

A fully specified **probability model** associates a numerical probability $P(\omega)$ with each possible world.[1] The basic axioms of probability theory say that every possible world has a probability between 0 and 1 and that the total probability of the set of possible worlds is 1:

$$0 \le P(\omega) \le 1 \text{ for every } \omega \text{ and } \sum_{\omega \in \Omega} P(\omega) = 1\ . \tag{13.1}$$

For example, if we assume that each die is fair and the rolls don't interfere with each other, then each of the possible worlds (1,1), (1,2), $\ldots$, (6,6) has probability 1/36. On the other hand, if the dice conspire to produce the same number, then the worlds (1,1), (2,2), (3,3), etc., might have higher probabilities, leaving the others with lower probabilities.

Probabilistic assertions and queries are not usually about particular possible worlds, but about sets of them. For example, we might be interested in the cases where the two dice add up to 11, the cases where doubles are rolled, and so on. In probability theory, these sets are called **events**—a term already used extensively in Chapter 12 for a different concept. In AI, the sets are always described by **propositions** in a formal language. (One such language is described in Section 13.2.2.) For each proposition, the corresponding set contains just those possible worlds in which the proposition holds. The probability associated with a proposition

EVENT

[1] For now, we assume a discrete, countable set of worlds. The proper treatment of the continuous case brings in certain complications that are less relevant for most purposes in AI.

is defined to be the sum of the probabilities of the worlds in which it holds:

$$\text{For any proposition } \phi,\ P(\phi) = \sum_{\omega \in \phi} P(\omega)\ . \tag{13.2}$$

For example, when rolling fair dice, we have $P(Total = 11) = P((5,6)) + P((6,5)) = 1/36 + 1/36 = 1/18$. Note that probability theory does not require complete knowledge of the probabilities of each possible world. For example, if we believe the dice conspire to produce the same number, we might *assert* that $P(doubles) = 1/4$ without knowing whether the dice prefer double 6 to double 2. Just as with logical assertions, this assertion *constrains* the underlying probability model without fully determining it.

UNCONDITIONAL PROBABILITY

PRIOR PROBABILITY

Probabilities such as $P(Total = 11)$ and $P(doubles)$ are called **unconditional** or **prior probabilities** (and sometimes just "priors" for short); they refer to degrees of belief in propositions *in the absence of any other information*. Most of the time, however, we have *some* information, usually called **evidence**, that has already been revealed. For example, the first die may already be showing a 5 and we are waiting with bated breath for the other one to stop spinning. In that case, we are interested not in the unconditional probability of rolling doubles, but the **conditional** or **posterior** probability (or just "posterior" for short) of rolling doubles *given that the first die is a 5*. This probability is written $P(doubles \mid Die_1 = 5)$, where the "$\mid$" is pronounced "given." Similarly, if I am going to the dentist for a regular checkup, the probability $P(cavity) = 0.2$ might be of interest; but if I go to the dentist because I have a toothache, it's $P(cavity \mid toothache) = 0.6$ that matters. Note that the precedence of "$\mid$" is such that any expression of the form $P(\ldots \mid \ldots)$ always means $P((\ldots)|(\ldots))$.

EVIDENCE

CONDITIONAL PROBABILITY

POSTERIOR PROBABILITY

It is important to understand that $P(cavity) = 0.2$ is still *valid* after $toothache$ is observed; it just isn't especially useful. When making decisions, an agent needs to condition on *all* the evidence it has observed. It is also important to understand the difference between conditioning and logical implication. The assertion that $P(cavity \mid toothache) = 0.6$ does not mean "Whenever $toothache$ is true, conclude that $cavity$ is true with probability 0.6" rather it means "Whenever $toothache$ is true *and we have no further information*, conclude that $cavity$ is true with probability 0.6." The extra condition is important; for example, if we had the further information that the dentist found no cavities, we definitely would not want to conclude that $cavity$ is true with probability 0.6; instead we need to use $P(cavity|toothache \wedge \neg cavity) = 0$.

Mathematically speaking, conditional probabilities are defined in terms of unconditional probabilities as follows: for any propositions $a$ and $b$, we have

$$P(a \mid b) = \frac{P(a \wedge b)}{P(b)}\ , \tag{13.3}$$

which holds whenever $P(b) > 0$. For example,

$$P(doubles \mid Die_1 = 5) = \frac{P(doubles \wedge Die_1 = 5)}{P(Die_1 = 5)}\ .$$

The definition makes sense if you remember that observing $b$ rules out all those possible worlds where $b$ is false, leaving a set whose total probability is just $P(b)$. Within that set, the $a$-worlds satisfy $a \wedge b$ and constitute a fraction $P(a \wedge b)/P(b)$.

The definition of conditional probability, Equation (13.3), can be written in a different form called the **product rule**:

PRODUCT RULE

$$P(a \land b) = P(a \mid b)P(b) ,$$

The product rule is perhaps easier to remember: it comes from the fact that, for $a$ and $b$ to be true, we need $b$ to be true, and we also need $a$ to be true given $b$.

### 13.2.2 The language of propositions in probability assertions

In this chapter and the next, propositions describing sets of possible worlds are written in a notation that combines elements of propositional logic and constraint satisfaction notation. In the terminology of Section 2.4.7, it is a **factored representation**, in which a possible world is represented by a set of variable/value pairs.

RANDOM VARIABLE

DOMAIN

Variables in probability theory are called **random variables** and their names begin with an uppercase letter. Thus, in the dice example, $Total$ and $Die_1$ are random variables. Every random variable has a **domain**—the set of possible values it can take on. The domain of $Total$ for two dice is the set $\{2, \ldots, 12\}$ and the domain of $Die_1$ is $\{1, \ldots, 6\}$. A Boolean random variable has the domain $\{true, false\}$ (notice that values are always lowercase); for example, the proposition that doubles are rolled can be written as $Doubles = true$. By convention, propositions of the form $A = true$ are abbreviated simply as $a$, while $A = false$ is abbreviated as $\neg a$. (The uses of $doubles$, $cavity$, and $toothache$ in the preceding section are abbreviations of this kind.) As in CSPs, domains can be sets of arbitrary tokens; we might choose the domain of $Age$ to be $\{juvenile, teen, adult\}$ and the domain of $Weather$ might be $\{sunny, rain, cloudy, snow\}$. When no ambiguity is possible, it is common to use a value by itself to stand for the proposition that a particular variable has that value; thus, $sunny$ can stand for $Weather = sunny$.

The preceding examples all have finite domains. Variables can have infinite domains, too—either discrete (like the integers) or continuous (like the reals). For any variable with an ordered domain, inequalities are also allowed, such as $NumberOfAtomsInUniverse \geq 10^{70}$.

Finally, we can combine these sorts of elementary propositions (including the abbreviated forms for Boolean variables) by using the connectives of propositional logic. For example, we can express "The probability that the patient has a cavity, given that she is a teenager with no toothache, is 0.1" as follows:

$$P(cavity \mid \neg toothache \land teen) = 0.1 .$$

Sometimes we will want to talk about the probabilities of *all* the possible values of a random variable. We could write:

$$\begin{aligned}
&P(Weather = sunny) = 0.6 \\
&P(Weather = rain) = 0.1 \\
&P(Weather = cloudy) = 0.29 \\
&P(Weather = snow) = 0.01 ,
\end{aligned}$$

but as an abbreviation we will allow

$$\mathbf{P}(Weather) = \langle 0.6, 0.1, 0.29, 0.01 \rangle ,$$

where the bold **P** indicates that the result is a vector of numbers, and where we assume a predefined ordering $\langle sunny, rain, cloudy, snow\rangle$ on the domain of $Weather$. We say that the **P** statement defines a **probability distribution** for the random variable $Weather$. The **P** notation is also used for conditional distributions: $\mathbf{P}(X \mid Y)$ gives the values of $P(X = x_i \mid Y = y_j)$ for each possible $i$, $j$ pair.

PROBABILITY DISTRIBUTION

For continuous variables, it is not possible to write out the entire distribution as a vector, because there are infinitely many values. Instead, we can define the probability that a random variable takes on some value $x$ as a parameterized function of $x$. For example, the sentence

$$P(NoonTemp = x) = Uniform_{[18C,26C]}(x)$$

expresses the belief that the temperature at noon is distributed uniformly between 18 and 26 degrees Celsius. We call this a **probability density function**.

PROBABILITY DENSITY FUNCTION

Probability density functions (sometimes called **pdfs**) differ in meaning from discrete distributions. Saying that the probability density is uniform from $18C$ to $26C$ means that there is a 100% chance that the temperature will fall somewhere in that $8C$-wide region and a 50% chance that it will fall in any $4C$-wide region, and so on. We write the probability density for a continuous random variable $X$ at value $x$ as $P(X = x)$ or just $P(x)$; the intuitive definition of $P(x)$ is the probability that $X$ falls within an arbitrarily small region beginning at $x$, divided by the width of the region:

$$P(x) = \lim_{dx\to 0} P(x \le X \le x + dx)/dx \ .$$

For $NoonTemp$ we have

$$P(NoonTemp = x) = Uniform_{[18C,26C]}(x) = \begin{cases} \frac{1}{8C} \text{ if } 18C \le x \le 26C \\ 0 \text{ otherwise} \end{cases} ,$$

where $C$ stands for centigrade (not for a constant). In $P(NoonTemp = 20.18C) = \frac{1}{8C}$, note that $\frac{1}{8C}$ is not a probability, it is a probability density. The probability that $NoonTemp$ is *exactly* $20.18C$ is zero, because $20.18C$ is a region of width 0. Some authors use different symbols for discrete distributions and density functions; we use $P$ in both cases, since confusion seldom arises and the equations are usually identical. Note that probabilities are unitless numbers, whereas density functions are measured with a unit, in this case reciprocal degrees.

In addition to distributions on single variables, we need notation for distributions on multiple variables. Commas are used for this. For example, $\mathbf{P}(Weather, Cavity)$ denotes the probabilities of all combinations of the values of $Weather$ and $Cavity$. This is a $4 \times 2$ table of probabilities called the **joint probability distribution** of $Weather$ and $Cavity$. We can also mix variables with and without values; $\mathbf{P}(sunny, Cavity)$ would be a two-element vector giving the probabilities of a sunny day with a cavity and a sunny day with no cavity. The **P** notation makes certain expressions much more concise than they might otherwise be. For example, the product rules for all possible values of $Weather$ and $Cavity$ can be written as a single equation:

JOINT PROBABILITY DISTRIBUTION

$$\mathbf{P}(Weather, Cavity) = \mathbf{P}(Weather \mid Cavity)\mathbf{P}(Cavity) \ ,$$

instead of as these $4 \times 2 = 8$ equations (using abbreviations $W$ and $C$):

$$\begin{aligned}
&P(W = sunny \wedge C = true) = P(W = sunny|C = true)\, P(C = true)\\
&P(W = rain \wedge C = true) = P(W = rain|C = true)\, P(C = true)\\
&P(W = cloudy \wedge C = true) = P(W = cloudy|C = true)\, P(C = true)\\
&P(W = snow \wedge C = true) = P(W = snow|C = true)\, P(C = true)\\
&P(W = sunny \wedge C = false) = P(W = sunny|C = false)\, P(C = false)\\
&P(W = rain \wedge C = false) = P(W = rain|C = false)\, P(C = false)\\
&P(W = cloudy \wedge C = false) = P(W = cloudy|C = false)\, P(C = false)\\
&P(W = snow \wedge C = false) = P(W = snow|C = false)\, P(C = false)\ .
\end{aligned}$$

As a degenerate case, $\mathbf{P}(sunny, cavity)$ has no variables and thus is a one-element vector that is the probability of a sunny day with a cavity, which could also be written as $P(sunny, cavity)$ or $P(sunny \wedge cavity)$. We will sometimes use **P** notation to derive results about individual $P$ values, and when we say "$\mathbf{P}(sunny) = 0.6$" it is really an abbreviation for "$\mathbf{P}(sunny)$ is the one-element vector $\langle 0.6 \rangle$, which means that $P(sunny) = 0.6$."

Now we have defined a syntax for propositions and probability assertions and we have given part of the semantics: Equation (13.2) defines the probability of a proposition as the sum of the probabilities of worlds in which it holds. To complete the semantics, we need to say what the worlds are and how to determine whether a proposition holds in a world. We borrow this part directly from the semantics of propositional logic, as follows. *A possible world is defined to be an assignment of values to all of the random variables under consideration.* It is easy to see that this definition satisfies the basic requirement that possible worlds be mutually exclusive and exhaustive (Exercise 13.5). For example, if the random variables are $Cavity$, $Toothache$, and $Weather$, then there are $2 \times 2 \times 4 = 16$ possible worlds. Furthermore, the truth of any given proposition, no matter how complex, can be determined easily in such worlds using the same recursive definition of truth as for formulas in propositional logic.

FULL JOINT PROBABILITY DISTRIBUTION

From the preceding definition of possible worlds, it follows that a probability model is completely determined by the joint distribution for all of the random variables—the so-called **full joint probability distribution**. For example, if the variables are $Cavity$, $Toothache$, and $Weather$, then the full joint distribution is given by $\mathbf{P}(Cavity, Toothache, Weather)$. This joint distribution can be represented as a $2 \times 2 \times 4$ table with 16 entries. Because every proposition's probability is a sum over possible worlds, a full joint distribution suffices, in principle, for calculating the probability of any proposition.

### 13.2.3 Probability axioms and their reasonableness

The basic axioms of probability (Equations (13.1) and (13.2)) imply certain relationships among the degrees of belief that can be accorded to logically related propositions. For example, we can derive the familiar relationship between the probability of a proposition and the probability of its negation:

$$\begin{aligned}
P(\neg a) &= \textstyle\sum_{\omega \in \neg a} P(\omega) && \text{by Equation (13.2)}\\
&= \textstyle\sum_{\omega \in \neg a} P(\omega) + \sum_{\omega \in a} P(\omega) - \sum_{\omega \in a} P(\omega) &&\\
&= \textstyle\sum_{\omega \in \Omega} P(\omega) - \sum_{\omega \in a} P(\omega) && \text{grouping the first two terms}\\
&= 1 - P(a) && \text{by (13.1) and (13.2).}
\end{aligned}$$

INCLUSION–EXCLUSION PRINCIPLE

We can also derive the well-known formula for the probability of a disjunction, sometimes called the **inclusion–exclusion principle**:

$$P(a \lor b) = P(a) + P(b) - P(a \land b) . \tag{13.4}$$

This rule is easily remembered by noting that the cases where $a$ holds, together with the cases where $b$ holds, certainly cover all the cases where $a \lor b$ holds; but summing the two sets of cases counts their intersection twice, so we need to subtract $P(a \land b)$. The proof is left as an exercise (Exercise 13.6).

KOLMOGOROV'S AXIOMS

Equations (13.1) and (13.4) are often called **Kolmogorov's axioms** in honor of the Russian mathematician Andrei Kolmogorov, who showed how to build up the rest of probability theory from this simple foundation and how to handle the difficulties caused by continuous variables.[2] While Equation (13.2) has a definitional flavor, Equation (13.4) reveals that the axioms really do constrain the degrees of belief an agent can have concerning logically related propositions. This is analogous to the fact that a logical agent cannot simultaneously believe $A$, $B$, and $\neg(A \land B)$, because there is no possible world in which all three are true. With probabilities, however, statements refer not to the world directly, but to the agent's own state of knowledge. Why, then, can an agent not hold the following set of beliefs (even though they violate Kolmogorov's axioms)?

$$\begin{array}{ll} P(a) = 0.4 & P(a \land b) = 0.0 \\ P(b) = 0.3 & P(a \lor b) = 0.8 . \end{array} \tag{13.5}$$

This kind of question has been the subject of decades of intense debate between those who advocate the use of probabilities as the only legitimate form for degrees of belief and those who advocate alternative approaches.

One argument for the axioms of probability, first stated in 1931 by Bruno de Finetti (and translated into English in de Finetti (1993)), is as follows: If an agent has some degree of belief in a proposition $a$, then the agent should be able to state odds at which it is indifferent to a bet for or against $a$.[3] Think of it as a game between two agents: Agent 1 states, "my degree of belief in event $a$ is 0.4." Agent 2 is then free to choose whether to wager for or against $a$ at stakes that are consistent with the stated degree of belief. That is, Agent 2 could choose to accept Agent 1's bet that $a$ will occur, offering \$6 against Agent 1's \$4. Or Agent 2 could accept Agent 1's bet that $\neg a$ will occur, offering \$4 against Agent 1's \$6. Then we observe the outcome of $a$, and whoever is right collects the money. If an agent's degrees of belief do not accurately reflect the world, then you would expect that it would tend to lose money over the long run to an opposing agent whose beliefs more accurately reflect the state of the world.

But de Finetti proved something much stronger: *If Agent 1 expresses a set of degrees of belief that violate the axioms of probability theory then there is a combination of bets by Agent 2 that guarantees that Agent 1 will lose money every time.* For example, suppose that Agent 1 has the set of degrees of belief from Equation (13.5). Figure 13.2 shows that if Agent

[2] The difficulties include the **Vitali set**, a well-defined subset of the interval $[0, 1]$ with no well-defined size.

[3] One might argue that the agent's preferences for different bank balances are such that the possibility of losing \$1 is not counterbalanced by an equal possibility of winning \$1. One possible response is to make the bet amounts small enough to avoid this problem. Savage's analysis (1954) circumvents the issue altogether.

2 chooses to bet \$4 on $a$, \$3 on $b$, and \$2 on $\neg(a \vee b)$, then Agent 1 always loses money, regardless of the outcomes for $a$ and $b$. De Finetti's theorem implies that no rational agent can have beliefs that violate the axioms of probability.

| Agent 1 | | Agent 2 | | Outcomes and payoffs to Agent 1 | | | |
|---|---|---|---|---|---|---|---|
| Proposition | Belief | Bet | Stakes | $a, b$ | $a, \neg b$ | $\neg a, b$ | $\neg a, \neg b$ |
| $a$ | 0.4 | $a$ | 4 to 6 | –6 | –6 | 4 | 4 |
| $b$ | 0.3 | $b$ | 3 to 7 | –7 | 3 | –7 | 3 |
| $a \vee b$ | 0.8 | $\neg(a \vee b)$ | 2 to 8 | 2 | 2 | 2 | –8 |
| | | | | –11 | –1 | –1 | –1 |

**Figure 13.2** Because Agent 1 has inconsistent beliefs, Agent 2 is able to devise a set of bets that guarantees a loss for Agent 1, no matter what the outcome of $a$ and $b$.

One common objection to de Finetti's theorem is that this betting game is rather contrived. For example, what if one refuses to bet? Does that end the argument? The answer is that the betting game is an abstract model for the decision-making situation in which every agent is *unavoidably* involved at every moment. Every action (including inaction) is a kind of bet, and every outcome can be seen as a payoff of the bet. Refusing to bet is like refusing to allow time to pass.

Other strong philosophical arguments have been put forward for the use of probabilities, most notably those of Cox (1946), Carnap (1950), and Jaynes (2003). They each construct a set of axioms for reasoning with degrees of beliefs: no contradictions, correspondence with ordinary logic (for example, if belief in $A$ goes up, then belief in $\neg A$ must go down), and so on. The only controversial axiom is that degrees of belief must be numbers, or at least act like numbers in that they must be transitive (if belief in $A$ is greater than belief in $B$, which is greater than belief in $C$, then belief in $A$ must be greater than $C$) and comparable (the belief in $A$ must be one of equal to, greater than, or less than belief in $B$). It can then be proved that probability is the only approach that satisfies these axioms.

The world being the way it is, however, practical demonstrations sometimes speak louder than proofs. The success of reasoning systems based on probability theory has been much more effective in making converts. We now look at how the axioms can be deployed to make inferences.

## 13.3 INFERENCE USING FULL JOINT DISTRIBUTIONS

PROBABILISTIC INFERENCE

In this section we describe a simple method for **probabilistic inference**—that is, the computation of posterior probabilities for query propositions given observed evidence. We use the full joint distribution as the "knowledge base" from which answers to all questions may be derived. Along the way we also introduce several useful techniques for manipulating equations involving probabilities.

## WHERE DO PROBABILITIES COME FROM?

There has been endless debate over the source and status of probability numbers. The **frequentist** position is that the numbers can come only from *experiments*: if we test 100 people and find that 10 of them have a cavity, then we can say that the probability of a cavity is approximately 0.1. In this view, the assertion "the probability of a cavity is 0.1" means that 0.1 is the fraction that would be observed in the limit of infinitely many samples. From any finite sample, we can estimate the true fraction and also calculate how accurate our estimate is likely to be.

The **objectivist** view is that probabilities are real aspects of the universe—propensities of objects to behave in certain ways—rather than being just descriptions of an observer's degree of belief. For example, the fact that a fair coin comes up heads with probability 0.5 is a propensity of the coin itself. In this view, frequentist measurements are attempts to observe these propensities. Most physicists agree that quantum phenomena are objectively probabilistic, but uncertainty at the macroscopic scale—e.g., in coin tossing—usually arises from ignorance of initial conditions and does not seem consistent with the propensity view.

The **subjectivist** view describes probabilities as a way of characterizing an agent's beliefs, rather than as having any external physical significance. The subjective **Bayesian** view allows any self-consistent ascription of prior probabilities to propositions, but then insists on proper Bayesian updating as evidence arrives.

In the end, even a strict frequentist position involves subjective analysis because of the **reference class** problem: in trying to determine the outcome probability of a *particular* experiment, the frequentist has to place it in a reference class of "similar" experiments with known outcome frequencies. I. J. Good (1983, p. 27) wrote, "every event in life is unique, and every real-life probability that we estimate in practice is that of an event that has never occurred before." For example, given a particular patient, a frequentist who wants to estimate the probability of a cavity will consider a reference class of other patients who are similar in important ways—age, symptoms, diet—and see what proportion of them had a cavity. If the dentist considers everything that is known about the patient—weight to the nearest gram, hair color, mother's maiden name—then the reference class becomes empty. This has been a vexing problem in the philosophy of science.

The **principle of indifference** attributed to Laplace (1816) states that propositions that are syntactically "symmetric" with respect to the evidence should be accorded equal probability. Various refinements have been proposed, culminating in the attempt by Carnap and others to develop a rigorous **inductive logic**, capable of computing the correct probability for any proposition from any collection of observations. Currently, it is believed that no unique inductive logic exists; rather, any such logic rests on a subjective prior probability distribution whose effect is diminished as more observations are collected.

| | *toothache* | | *¬toothache* | |
|---|---|---|---|---|
| | *catch* | *¬catch* | *catch* | *¬catch* |
| *cavity* | 0.108 | 0.012 | 0.072 | 0.008 |
| *¬cavity* | 0.016 | 0.064 | 0.144 | 0.576 |

**Figure 13.3** A full joint distribution for the $Toothache$, $Cavity$, $Catch$ world.

We begin with a simple example: a domain consisting of just the three Boolean variables $Toothache$, $Cavity$, and $Catch$ (the dentist's nasty steel probe catches in my tooth). The full joint distribution is a $2 \times 2 \times 2$ table as shown in Figure 13.3.

Notice that the probabilities in the joint distribution sum to 1, as required by the axioms of probability. Notice also that Equation (13.2) gives us a direct way to calculate the probability of any proposition, simple or complex: simply identify those possible worlds in which the proposition is true and add up their probabilities. For example, there are six possible worlds in which $cavity \lor toothache$ holds:

$$P(cavity \lor toothache) = 0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28\ .$$

One particularly common task is to extract the distribution over some subset of variables or a single variable. For example, adding the entries in the first row gives the unconditional or

MARGINAL PROBABILITY

**marginal probability**[4] of $cavity$:

$$P(cavity) = 0.108 + 0.012 + 0.072 + 0.008 = 0.2\ .$$

MARGINALIZATION

This process is called **marginalization**, or **summing out**—because we sum up the probabilities for each possible value of the other variables, thereby taking them out of the equation. We can write the following general marginalization rule for any sets of variables **Y** and **Z**:

$$\mathbf{P}(\mathbf{Y}) = \sum_{\mathbf{z}\in\mathbf{Z}} \mathbf{P}(\mathbf{Y}, \mathbf{z})\ , \tag{13.6}$$

where $\sum_{\mathbf{z}\in\mathbf{Z}}$ means to sum over all the possible combinations of values of the set of variables **Z**. We sometimes abbreviate this as $\sum_{\mathbf{z}}$, leaving **Z** implicit. We just used the rule as

$$\mathbf{P}(Cavity) = \sum_{\mathbf{z}\in\{Catch, Toothache\}} \mathbf{P}(Cavity, \mathbf{z})\ . \tag{13.7}$$

A variant of this rule involves conditional probabilities instead of joint probabilities, using the product rule:

$$\mathbf{P}(\mathbf{Y}) = \sum_{\mathbf{z}} \mathbf{P}(\mathbf{Y} \mid \mathbf{z}) P(\mathbf{z})\ . \tag{13.8}$$

CONDITIONING

This rule is called **conditioning**. Marginalization and conditioning turn out to be useful rules for all kinds of derivations involving probability expressions.

In most cases, we are interested in computing *conditional* probabilities of some variables, given evidence about others. Conditional probabilities can be found by first using

[4] So called because of a common practice among actuaries of writing the sums of observed frequencies in the margins of insurance tables.

Equation (13.3) to obtain an expression in terms of unconditional probabilities and then evaluating the expression from the full joint distribution. For example, we can compute the probability of a cavity, given evidence of a toothache, as follows:

$$\begin{aligned} P(cavity \mid toothache) &= \frac{P(cavity \wedge toothache)}{P(toothache)} \\ &= \frac{0.108 + 0.012}{0.108 + 0.012 + 0.016 + 0.064} = 0.6 \,. \end{aligned}$$

Just to check, we can also compute the probability that there is no cavity, given a toothache:

$$\begin{aligned} P(\neg cavity \mid toothache) &= \frac{P(\neg cavity \wedge toothache)}{P(toothache)} \\ &= \frac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064} = 0.4 \,. \end{aligned}$$

The two values sum to 1.0, as they should. Notice that in these two calculations the term $1/P(toothache)$ remains constant, no matter which value of $Cavity$ we calculate. In fact, it can be viewed as a **normalization** constant for the distribution $\mathbf{P}(Cavity \mid toothache)$, ensuring that it adds up to 1. Throughout the chapters dealing with probability, we use $\alpha$ to denote such constants. With this notation, we can write the two preceding equations in one:

NORMALIZATION

$$\begin{aligned} &\mathbf{P}(Cavity \mid toothache) = \alpha\, \mathbf{P}(Cavity, toothache) \\ &\quad = \alpha\, [\mathbf{P}(Cavity, toothache, catch) + \mathbf{P}(Cavity, toothache, \neg catch)] \\ &\quad = \alpha\, [\langle 0.108, 0.016\rangle + \langle 0.012, 0.064\rangle] = \alpha\, \langle 0.12, 0.08\rangle = \langle 0.6, 0.4\rangle \,. \end{aligned}$$

In other words, we can calculate $\mathbf{P}(Cavity \mid toothache)$ even if we don't know the value of $P(toothache)$! We temporarily forget about the factor $1/P(toothache)$ and add up the values for $cavity$ and $\neg cavity$, getting 0.12 and 0.08. Those are the correct relative proportions, but they don't sum to 1, so we normalize them by dividing each one by $0.12 + 0.08$, getting the true probabilities of 0.6 and 0.4. Normalization turns out to be a useful shortcut in many probability calculations, both to make the computation easier and to allow us to proceed when some probability assessment (such as $P(toothache)$) is not available.

From the example, we can extract a general inference procedure. We begin with the case in which the query involves a single variable, $X$ ($Cavity$ in the example). Let $\mathbf{E}$ be the list of evidence variables (just $Toothache$ in the example), let $\mathbf{e}$ be the list of observed values for them, and let $\mathbf{Y}$ be the remaining unobserved variables (just $Catch$ in the example). The query is $\mathbf{P}(X \mid \mathbf{e})$ and can be evaluated as

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha\, \mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y}) \,, \tag{13.9}$$

where the summation is over all possible $\mathbf{y}$s (i.e., all possible combinations of values of the unobserved variables $\mathbf{Y}$). Notice that together the variables $X$, $\mathbf{E}$, and $\mathbf{Y}$ constitute the complete set of variables for the domain, so $\mathbf{P}(X, \mathbf{e}, \mathbf{y})$ is simply a subset of probabilities from the full joint distribution.

Given the full joint distribution to work with, Equation (13.9) can answer probabilistic queries for discrete variables. It does not scale well, however: for a domain described by $n$ Boolean variables, it requires an input table of size $O(2^n)$ and takes $O(2^n)$ time to process the table. In a realistic problem we could easily have $n > 100$, making $O(2^n)$ impractical. The full joint distribution in tabular form is just not a practical tool for building reasoning systems. Instead, it should be viewed as the theoretical foundation on which more effective approaches may be built, just as truth tables formed a theoretical foundation for more practical algorithms like DPLL. The remainder of this chapter introduces some of the basic ideas required in preparation for the development of realistic systems in Chapter 14.

## 13.4 INDEPENDENCE

Let us expand the full joint distribution in Figure 13.3 by adding a fourth variable, $Weather$. The full joint distribution then becomes $\mathbf{P}(Toothache, Catch, Cavity, Weather)$, which has $2 \times 2 \times 2 \times 4 = 32$ entries. It contains four "editions" of the table shown in Figure 13.3, one for each kind of weather. What relationship do these editions have to each other and to the original three-variable table? For example, how are $P(toothache, catch, cavity, cloudy)$ and $P(toothache, catch, cavity)$ related? We can use the product rule:

$$\begin{aligned}&P(toothache, catch, cavity, cloudy)\\ &\quad = P(cloudy \mid toothache, catch, cavity)P(toothache, catch, cavity)\ .\end{aligned}$$

Now, unless one is in the deity business, one should not imagine that one's dental problems influence the weather. And for indoor dentistry, at least, it seems safe to say that the weather does not influence the dental variables. Therefore, the following assertion seems reasonable:

$$P(cloudy \mid toothache, catch, cavity) = P(cloudy)\ . \tag{13.10}$$

From this, we can deduce

$$P(toothache, catch, cavity, cloudy) = P(cloudy)P(toothache, catch, cavity)\ .$$

A similar equation exists for *every entry* in $\mathbf{P}(Toothache, Catch, Cavity, Weather)$. In fact, we can write the general equation

$$\mathbf{P}(Toothache, Catch, Cavity, Weather) = \mathbf{P}(Toothache, Catch, Cavity)\mathbf{P}(Weather)\ .$$

Thus, the 32-element table for four variables can be constructed from one 8-element table and one 4-element table. This decomposition is illustrated schematically in Figure 13.4(a).

INDEPENDENCE

The property we used in Equation (13.10) is called **independence** (also **marginal independence** and **absolute independence**). In particular, the weather is independent of one's dental problems. Independence between propositions $a$ and $b$ can be written as

$$P(a \mid b) = P(a) \quad \text{or} \quad P(b \mid a) = P(b) \quad \text{or} \quad P(a \wedge b) = P(a)P(b)\ . \tag{13.11}$$

All these forms are equivalent (Exercise 13.12). Independence between variables $X$ and $Y$ can be written as follows (again, these are all equivalent):

$$\mathbf{P}(X \mid Y) = \mathbf{P}(X) \quad \text{or} \quad \mathbf{P}(Y \mid X) = \mathbf{P}(Y) \quad \text{or} \quad \mathbf{P}(X, Y) = \mathbf{P}(X)\mathbf{P}(Y)\ .$$

Independence assertions are usually based on knowledge of the domain. As the toothache–weather example illustrates, they can dramatically reduce the amount of information necessary to specify the full joint distribution. If the complete set of variables can be divided

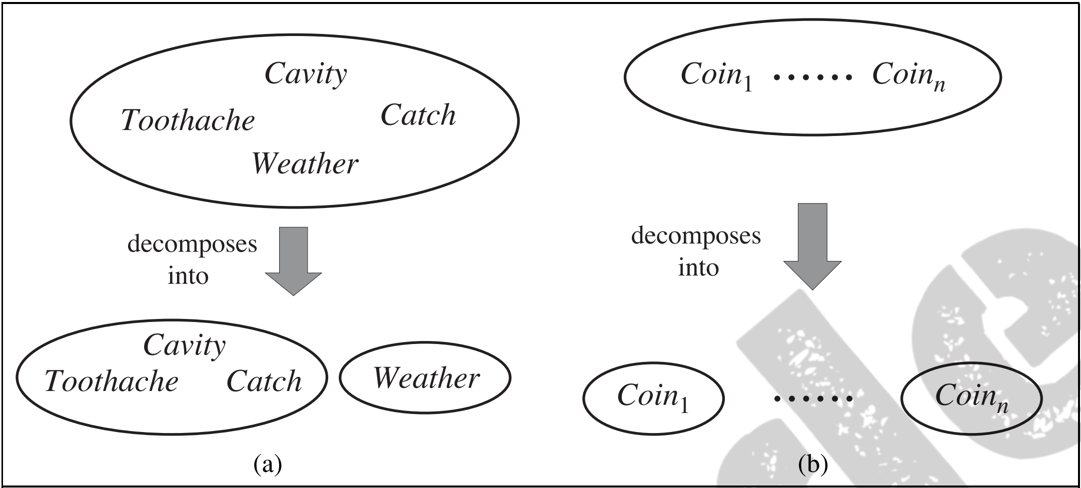

**Figure 13.4** Two examples of factoring a large joint distribution into smaller distributions, using absolute independence. (a) Weather and dental problems are independent. (b) Coin flips are independent.

into independent subsets, then the full joint distribution can be *factored* into separate joint distributions on those subsets. For example, the full joint distribution on the outcome of $n$ independent coin flips, $\mathbf{P}(C_1, \ldots, C_n)$, has $2^n$ entries, but it can be represented as the product of $n$ single-variable distributions $\mathbf{P}(C_i)$. In a more practical vein, the independence of dentistry and meteorology is a good thing, because otherwise the practice of dentistry might require intimate knowledge of meteorology, and vice versa.

When they are available, then, independence assertions can help in reducing the size of the domain representation and the complexity of the inference problem. Unfortunately, clean separation of entire sets of variables by independence is quite rare. Whenever a connection, however indirect, exists between two variables, independence will fail to hold. Moreover, even independent subsets can be quite large—for example, dentistry might involve dozens of diseases and hundreds of symptoms, all of which are interrelated. To handle such problems, we need more subtle methods than the straightforward concept of independence.

## 13.5 Bayes' Rule and Its Use

On page 486, we defined the **product rule**. It can actually be written in two forms:

$$P(a \land b) = P(a \mid b)P(b) \qquad \text{and} \qquad P(a \land b) = P(b \mid a)P(a) .$$

Equating the two right-hand sides and dividing by $P(a)$, we get

$$P(b \mid a) = \frac{P(a \mid b)P(b)}{P(a)} . \qquad (13.12)$$

BAYES' RULE

This equation is known as **Bayes' rule** (also Bayes' law or Bayes' theorem). This simple equation underlies most modern AI systems for probabilistic inference.

The more general case of Bayes' rule for multivalued variables can be written in the **P** notation as follows:

$$\mathbf{P}(Y \mid X) = \frac{\mathbf{P}(X \mid Y)\mathbf{P}(Y)}{\mathbf{P}(X)} ,$$

As before, this is to be taken as representing a set of equations, each dealing with specific values of the variables. We will also have occasion to use a more general version conditionalized on some background evidence **e**:

$$\mathbf{P}(Y \mid X, \mathbf{e}) = \frac{\mathbf{P}(X \mid Y, \mathbf{e})\mathbf{P}(Y \mid \mathbf{e})}{\mathbf{P}(X \mid \mathbf{e})} . \tag{13.13}$$

### 13.5.1 Applying Bayes' rule: The simple case

On the surface, Bayes' rule does not seem very useful. It allows us to compute the single term $P(b \mid a)$ in terms of three terms: $P(a \mid b)$, $P(b)$, and $P(a)$. That seems like two steps backwards, but Bayes' rule is useful in practice because there are many cases where we do have good probability estimates for these three numbers and need to compute the fourth. Often, we perceive as evidence the *effect* of some unknown *cause* and we would like to determine that cause. In that case, Bayes' rule becomes

$$P(cause \mid effect) = \frac{P(effect \mid cause)P(cause)}{P(effect)} .$$

CAUSAL

DIAGNOSTIC

The conditional probability $P(effect \mid cause)$ quantifies the relationship in the **causal** direction, whereas $P(cause \mid effect)$ describes the **diagnostic** direction. In a task such as medical diagnosis, we often have conditional probabilities on causal relationships (that is, the doctor knows $P(symptoms \mid disease)$) and want to derive a diagnosis, $P(disease \mid symptoms)$. For example, a doctor knows that the disease meningitis causes the patient to have a stiff neck, say, 70% of the time. The doctor also knows some unconditional facts: the prior probability that a patient has meningitis is 1/50,000, and the prior probability that any patient has a stiff neck is 1%. Letting $s$ be the proposition that the patient has a stiff neck and $m$ be the proposition that the patient has meningitis, we have

$$\begin{aligned} P(s \mid m) &= 0.7 \\ P(m) &= 1/50000 \\ P(s) &= 0.01 \\ P(m \mid s) &= \frac{P(s \mid m)P(m)}{P(s)} = \frac{0.7 \times 1/50000}{0.01} = 0.0014 . \end{aligned} \tag{13.14}$$

That is, we expect less than 1 in 700 patients with a stiff neck to have meningitis. Notice that even though a stiff neck is quite strongly indicated by meningitis (with probability 0.7), the probability of meningitis in the patient remains small. This is because the prior probability of stiff necks is much higher than that of meningitis.

Section 13.3 illustrated a process by which one can avoid assessing the prior probability of the evidence (here, $P(s)$) by instead computing a posterior probability for each value of the query variable (here, $m$ and $\neg m$) and then normalizing the results. The same process can be applied when using Bayes' rule. We have

$$\mathbf{P}(M \mid s) = \alpha \langle P(s \mid m)P(m), P(s \mid \neg m)P(\neg m)\rangle \ .$$

Thus, to use this approach we need to estimate $P(s \mid \neg m)$ instead of $P(s)$. There is no free lunch—sometimes this is easier, sometimes it is harder. The general form of Bayes' rule with normalization is

$$\mathbf{P}(Y \mid X) = \alpha\, \mathbf{P}(X \mid Y)\mathbf{P}(Y) \ , \qquad (13.15)$$

where $\alpha$ is the normalization constant needed to make the entries in $\mathbf{P}(Y \mid X)$ sum to 1.

One obvious question to ask about Bayes' rule is why one might have available the conditional probability in one direction, but not the other. In the meningitis domain, perhaps the doctor knows that a stiff neck implies meningitis in 1 out of 5000 cases; that is, the doctor has quantitative information in the **diagnostic** direction from symptoms to causes. Such a doctor has no need to use Bayes' rule. Unfortunately, *diagnostic knowledge is often more fragile than causal knowledge.* If there is a sudden epidemic of meningitis, the unconditional probability of meningitis, $P(m)$, will go up. The doctor who derived the diagnostic probability $P(m \mid s)$ directly from statistical observation of patients before the epidemic will have no idea how to update the value, but the doctor who computes $P(m \mid s)$ from the other three values will see that $P(m \mid s)$ should go up proportionately with $P(m)$. Most important, the causal information $P(s \mid m)$ is *unaffected* by the epidemic, because it simply reflects the way meningitis works. The use of this kind of direct causal or model-based knowledge provides the crucial robustness needed to make probabilistic systems feasible in the real world.

### 13.5.2 Using Bayes' rule: Combining evidence

We have seen that Bayes' rule can be useful for answering probabilistic queries conditioned on one piece of evidence—for example, the stiff neck. In particular, we have argued that probabilistic information is often available in the form $P(\mathit{effect} \mid \mathit{cause})$. What happens when we have two or more pieces of evidence? For example, what can a dentist conclude if her nasty steel probe catches in the aching tooth of a patient? If we know the full joint distribution (Figure 13.3), we can read off the answer:

$$\mathbf{P}(\mathit{Cavity} \mid \mathit{toothache} \wedge \mathit{catch}) = \alpha\,\langle 0.108, 0.016\rangle \approx \langle 0.871, 0.129\rangle \ .$$

We know, however, that such an approach does not scale up to larger numbers of variables. We can try using Bayes' rule to reformulate the problem:

$$\begin{aligned}&\mathbf{P}(\mathit{Cavity} \mid \mathit{toothache} \wedge \mathit{catch})\\ &\quad = \alpha\,\mathbf{P}(\mathit{toothache} \wedge \mathit{catch} \mid \mathit{Cavity})\,\mathbf{P}(\mathit{Cavity}) \ . \qquad (13.16)\end{aligned}$$

For this reformulation to work, we need to know the conditional probabilities of the conjunction $\mathit{toothache} \wedge \mathit{catch}$ for each value of $\mathit{Cavity}$. That might be feasible for just two evidence variables, but again it does not scale up. If there are $n$ possible evidence variables (X rays, diet, oral hygiene, etc.), then there are $2^n$ possible combinations of observed values for which we would need to know conditional probabilities. We might as well go back to using the full joint distribution. This is what first led researchers away from probability theory toward approximate methods for evidence combination that, while giving incorrect answers, require fewer numbers to give any answer at all.

Rather than taking this route, we need to find some additional assertions about the domain that will enable us to simplify the expressions. The notion of **independence** in Section 13.4 provides a clue, but needs refining. It would be nice if $Toothache$ and $Catch$ were independent, but they are not: if the probe catches in the tooth, then it is likely that the tooth has a cavity and that the cavity causes a toothache. These variables *are* independent, however, *given the presence or the absence of a cavity*. Each is directly caused by the cavity, but neither has a direct effect on the other: toothache depends on the state of the nerves in the tooth, whereas the probe's accuracy depends on the dentist's skill, to which the toothache is irrelevant.[5] Mathematically, this property is written as

$$\mathbf{P}(toothache \wedge catch \mid Cavity) = \mathbf{P}(toothache \mid Cavity)\mathbf{P}(catch \mid Cavity)\ . \quad (13.17)$$

CONDITIONAL INDEPENDENCE

This equation expresses the **conditional independence** of $toothache$ and $catch$ given $Cavity$. We can plug it into Equation (13.16) to obtain the probability of a cavity:

$$\begin{aligned}&\mathbf{P}(Cavity \mid toothache \wedge catch)\\ &\quad = \alpha\, \mathbf{P}(toothache \mid Cavity)\, \mathbf{P}(catch \mid Cavity)\, \mathbf{P}(Cavity)\ . \end{aligned} \quad (13.18)$$

Now the information requirements are the same as for inference, using each piece of evidence separately: the prior probability $\mathbf{P}(Cavity)$ for the query variable and the conditional probability of each effect, given its cause.

The general definition of **conditional independence** of two variables $X$ and $Y$, given a third variable $Z$, is

$$\mathbf{P}(X, Y \mid Z) = \mathbf{P}(X \mid Z)\mathbf{P}(Y \mid Z)\ .$$

In the dentist domain, for example, it seems reasonable to assert conditional independence of the variables $Toothache$ and $Catch$, given $Cavity$:

$$\mathbf{P}(Toothache, Catch \mid Cavity) = \mathbf{P}(Toothache \mid Cavity)\mathbf{P}(Catch \mid Cavity)\ . \quad (13.19)$$

Notice that this assertion is somewhat stronger than Equation (13.17), which asserts independence only for specific values of $Toothache$ and $Catch$. As with absolute independence in Equation (13.11), the equivalent forms

$$\mathbf{P}(X \mid Y, Z) = \mathbf{P}(X \mid Z) \quad \text{and} \quad \mathbf{P}(Y \mid X, Z) = \mathbf{P}(Y \mid Z)$$

can also be used (see Exercise 13.17). Section 13.4 showed that absolute independence assertions allow a decomposition of the full joint distribution into much smaller pieces. It turns out that the same is true for conditional independence assertions. For example, given the assertion in Equation (13.19), we can derive a decomposition as follows:

$$\begin{aligned}&\mathbf{P}(Toothache, Catch, Cavity)\\ &\quad = \mathbf{P}(Toothache, Catch \mid Cavity)\mathbf{P}(Cavity) \quad \text{(product rule)}\\ &\quad = \mathbf{P}(Toothache \mid Cavity)\mathbf{P}(Catch \mid Cavity)\mathbf{P}(Cavity) \quad \text{(using 13.19).}\end{aligned}$$

(The reader can easily check that this equation does in fact hold in Figure 13.3.) In this way, the original large table is decomposed into three smaller tables. The original table has seven

[5] We assume that the patient and dentist are distinct individuals.

independent numbers ($2^3 = 8$ entries in the table, but they must sum to 1, so 7 are independent). The smaller tables contain five independent numbers (for a conditional probability distributions such as $\mathbf{P}(T|C$ there are two rows of two numbers, and each row sums to 1, so that's two independent numbers; for a prior distribution like $\mathbf{P}(C)$ there is only one independent number). Going from seven to five might not seem like a major triumph, but the point is that, for $n$ symptoms that are all conditionally independent given $Cavity$, the size of the representation grows as $O(n)$ instead of $O(2^n)$. That means that *conditional independence assertions can allow probabilistic systems to scale up; moreover, they are much more commonly available than absolute independence assertions.* Conceptually, $Cavity$ **separates** $Toothache$ and $Catch$ because it is a direct cause of both of them. The decomposition of large probabilistic domains into weakly connected subsets through conditional independence is one of the most important developments in the recent history of AI.

SEPARATION

The dentistry example illustrates a commonly occurring pattern in which a single cause directly influences a number of effects, all of which are conditionally independent, given the cause. The full joint distribution can be written as

$$\mathbf{P}(Cause, Effect_1, \ldots, Effect_n) = \mathbf{P}(Cause) \prod_i \mathbf{P}(Effect_i \mid Cause) .$$

NAIVE BAYES

Such a probability distribution is called a **naive Bayes** model—"naive" because it is often used (as a simplifying assumption) in cases where the "effect" variables are *not* actually conditionally independent given the cause variable. (The naive Bayes model is sometimes called a **Bayesian classifier**, a somewhat careless usage that has prompted true Bayesians to call it the **idiot Bayes** model.) In practice, naive Bayes systems can work surprisingly well, even when the conditional independence assumption is not true. Chapter 20 describes methods for learning naive Bayes distributions from observations.

## 13.6 THE WUMPUS WORLD REVISITED

We can combine of the ideas in this chapter to solve probabilistic reasoning problems in the wumpus world. (See Chapter 7 for a complete description of the wumpus world.) Uncertainty arises in the wumpus world because the agent's sensors give only partial information about the world. For example, Figure 13.5 shows a situation in which each of the three reachable squares—[1,3], [2,2], and [3,1]—might contain a pit. Pure logical inference can conclude nothing about which square is most likely to be safe, so a logical agent might have to choose randomly. We will see that a probabilistic agent can do much better than the logical agent.

Our aim is to calculate the probability that each of the three squares contains a pit. (For this example we ignore the wumpus and the gold.) The relevant properties of the wumpus world are that (1) a pit causes breezes in all neighboring squares, and (2) each square other than [1,1] contains a pit with probability 0.2. The first step is to identify the set of random variables we need:

- As in the propositional logic case, we want one Boolean variable $P_{ij}$ for each square, which is true iff square $[i,j]$ actually contains a pit.

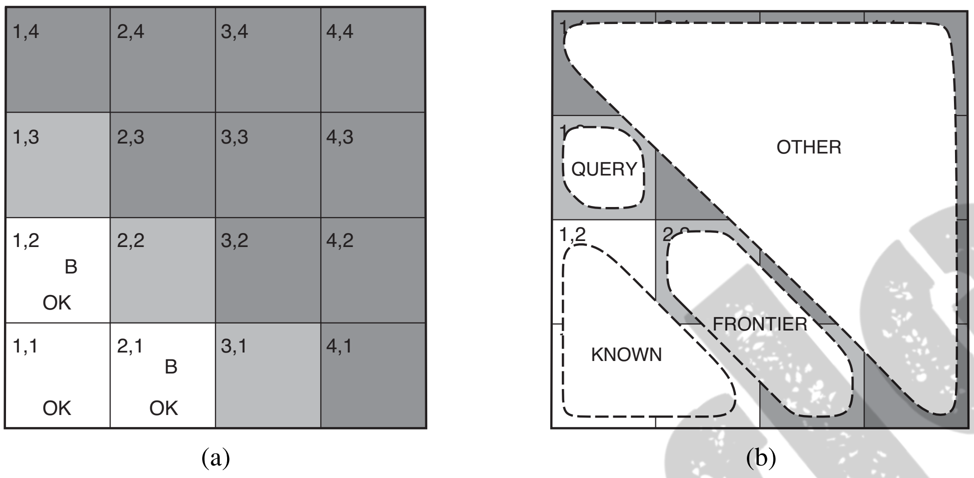

**Figure 13.5** (a) After finding a breeze in both [1,2] and [2,1], the agent is stuck—there is no safe place to explore. (b) Division of the squares into *Known*, *Frontier*, and *Other*, for a query about [1,3].

- We also have Boolean variables $B_{ij}$ that are true iff square $[i, j]$ is breezy; we include these variables only for the observed squares—in this case, [1,1], [1,2], and [2,1].

The next step is to specify the full joint distribution, $\mathbf{P}(P_{1,1}, \ldots, P_{4,4}, B_{1,1}, B_{1,2}, B_{2,1})$. Applying the product rule, we have

$$\mathbf{P}(P_{1,1}, \ldots, P_{4,4}, B_{1,1}, B_{1,2}, B_{2,1}) = \\ \mathbf{P}(B_{1,1}, B_{1,2}, B_{2,1} \mid P_{1,1}, \ldots, P_{4,4})\mathbf{P}(P_{1,1}, \ldots, P_{4,4}) .$$

This decomposition makes it easy to see what the joint probability values should be. The first term is the conditional probability distribution of a breeze configuration, given a pit configuration; its values are 1 if the breezes are adjacent to the pits and 0 otherwise. The second term is the prior probability of a pit configuration. Each square contains a pit with probability 0.2, independently of the other squares; hence,

$$\mathbf{P}(P_{1,1}, \ldots, P_{4,4}) = \prod_{i,j=1,1}^{4,4} \mathbf{P}(P_{i,j}) . \quad (13.20)$$

For a particular configuration with exactly $n$ pits, $P(P_{1,1}, \ldots, P_{4,4}) = 0.2^n \times 0.8^{16-n}$.

In the situation in Figure 13.5(a), the evidence consists of the observed breeze (or its absence) in each square that is visited, combined with the fact that each such square contains no pit. We abbreviate these facts as $b = \neg b_{1,1} \wedge b_{1,2} \wedge b_{2,1}$ and $known = \neg p_{1,1} \wedge \neg p_{1,2} \wedge \neg p_{2,1}$. We are interested in answering queries such as $\mathbf{P}(P_{1,3} \mid known, b)$: how likely is it that [1,3] contains a pit, given the observations so far?

To answer this query, we can follow the standard approach of Equation (13.9), namely, summing over entries from the full joint distribution. Let *Unknown* be the set of $P_{i,j}$ variables for squares other than the *Known* squares and the query square [1,3]. Then, by Equation (13.9), we have

$$\mathbf{P}(P_{1,3} \mid known, b) = \alpha \sum_{unknown} \mathbf{P}(P_{1,3}, unknown, known, b) \,.$$

The full joint probabilities have already been specified, so we are done—that is, unless we care about computation. There are 12 unknown squares; hence the summation contains $2^{12} = 4096$ terms. In general, the summation grows exponentially with the number of squares.

Surely, one might ask, aren't the other squares irrelevant? How could [4,4] affect whether [1,3] has a pit? Indeed, this intuition is correct. Let *Frontier* be the pit variables (other than the query variable) that are adjacent to visited squares, in this case just [2,2] and [3,1]. Also, let *Other* be the pit variables for the other unknown squares; in this case, there are 10 other squares, as shown in Figure 13.5(b). The key insight is that the observed breezes are *conditionally independent* of the other variables, given the known, frontier, and query variables. To use the insight, we manipulate the query formula into a form in which the breezes are conditioned on all the other variables, and then we apply conditional independence:

$$\begin{aligned}
&\mathbf{P}(P_{1,3} \mid known, b)\\
&\quad = \alpha \sum_{unknown} \mathbf{P}(P_{1,3}, known, b, unknown) \qquad \text{(by Equation (13.9))}\\
&\quad = \alpha \sum_{unknown} \mathbf{P}(b \mid P_{1,3}, known, unknown)\mathbf{P}(P_{1,3}, known, unknown)\\
&\qquad\qquad\qquad\qquad \text{(by the product rule)}\\
&\quad = \alpha \sum_{frontier} \sum_{other} \mathbf{P}(b \mid known, P_{1,3}, frontier, other)\mathbf{P}(P_{1,3}, known, frontier, other)\\
&\quad = \alpha \sum_{frontier} \sum_{other} \mathbf{P}(b \mid known, P_{1,3}, frontier)\mathbf{P}(P_{1,3}, known, frontier, other)\,,
\end{aligned}$$

where the final step uses conditional independence: $b$ is independent of *other* given *known*, $P_{1,3}$, and *frontier*. Now, the first term in this expression does not depend on the *Other* variables, so we can move the summation inward:

$$\begin{aligned}
&\mathbf{P}(P_{1,3} \mid known, b)\\
&\quad = \alpha \sum_{frontier} \mathbf{P}(b \mid known, P_{1,3}, frontier) \sum_{other} \mathbf{P}(P_{1,3}, known, frontier, other)\,.
\end{aligned}$$

By independence, as in Equation (13.20), the prior term can be factored, and then the terms can be reordered:

$$\begin{aligned}
&\mathbf{P}(P_{1,3} \mid known, b)\\
&\quad = \alpha \sum_{frontier} \mathbf{P}(b \mid known, P_{1,3}, frontier) \sum_{other} \mathbf{P}(P_{1,3})P(known)P(frontier)P(other)\\
&\quad = \alpha\, P(known)\mathbf{P}(P_{1,3}) \sum_{frontier} \mathbf{P}(b \mid known, P_{1,3}, frontier)P(frontier) \sum_{other} P(other)\\
&\quad = \alpha'\, \mathbf{P}(P_{1,3}) \sum_{frontier} \mathbf{P}(b \mid known, P_{1,3}, frontier)P(frontier)\,,
\end{aligned}$$

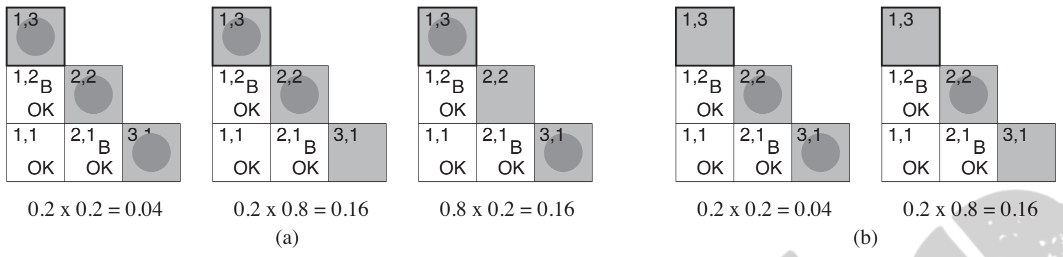

**Figure 13.6** Consistent models for the frontier variables $P_{2,2}$ and $P_{3,1}$, showing $P(frontier)$ for each model: (a) three models with $P_{1,3} = true$ showing two or three pits, and (b) two models with $P_{1,3} = false$ showing one or two pits.

where the last step folds $P(known)$ into the normalizing constant and uses the fact that $\sum_{other} P(other)$ equals 1.

Now, there are just four terms in the summation over the frontier variables $P_{2,2}$ and $P_{3,1}$. The use of independence and conditional independence has completely eliminated the other squares from consideration.

Notice that the expression $\mathbf{P}(b \mid known, P_{1,3}, frontier)$ is 1 when the frontier is consistent with the breeze observations, and 0 otherwise. Thus, for each value of $P_{1,3}$, we sum over the *logical models* for the frontier variables that are consistent with the known facts. (Compare with the enumeration over models in Figure 7.5 on page 241.) The models and their associated prior probabilities—$P(frontier)$—are shown in Figure 13.6. We have

$$\mathbf{P}(P_{1,3} \mid known, b) = \alpha' \langle 0.2(0.04 + 0.16 + 0.16),\ 0.8(0.04 + 0.16) \rangle \approx \langle 0.31, 0.69 \rangle .$$

That is, [1,3] (and [3,1] by symmetry) contains a pit with roughly 31% probability. A similar calculation, which the reader might wish to perform, shows that [2,2] contains a pit with roughly 86% probability. The wumpus agent should definitely avoid [2,2]! Note that our logical agent from Chapter 7 did not know that [2,2] was worse than the other squares. Logic can tell us that it is unknown whether there is a pit in [2, 2], but we need probability to tell us how likely it is.

What this section has shown is that even seemingly complicated problems can be formulated precisely in probability theory and solved with simple algorithms. To get *efficient* solutions, independence and conditional independence relationships can be used to simplify the summations required. These relationships often correspond to our natural understanding of how the problem should be decomposed. In the next chapter, we develop formal representations for such relationships as well as algorithms that operate on those representations to perform probabilistic inference efficiently.

# EXPERT SYSTEMS

*An expert is a man who has made all the mistakes which can be made in a very narrow field.*

**—Niels Bohr**
(1885–1962), Danish physicist

Expert systems solve problems (such as the ones in Fig. 1.1) that are normally solved by human "experts." To solve expert-level problems, expert systems need access to a substantial domain knowledge base, which must be built as efficiently as possible. They also need to exploit one or more reasoning mechanisms to apply their knowledge to the problems they are given. Then they need a mechanism for explaining what they have done to the users who rely on them. One way to look at expert systems is that they represent applied AI in a very broad sense. They tend to lag several years behind research advances, but because they are tackling harder and harder problems, they will eventually be able to make use of all of the kinds of results that we have described throughout this book. So this chapter is in some ways a review of much of what we have already discussed.

The problems that expert systems deal with are highly diverse. There are some general issues that arise across these varying domains. But it also turns out that there are powerful techniques that can be defined for specific classes of problems. Recall that in Section 2.3.8 we introduced the notion of problem classification and we described some classes into which problems can be organized. Throughout this chapter we have occasion to return to this idea, and we see how some key problem characteristics play an important role in guiding the design of problem-solving systems. For example, it is now clear that tools that are developed to support one classification or diagnosis task are often useful for another, while different tools are useful for solving various kinds of design tasks.

## 20.1 REPRESENTING AND USING DOMAIN KNOWLEDGE

Expert systems are complex AI programs. Almost all the techniques that we described in Parts I and II have been exploited in at least one expert system. However, the most widely used way of representing domain knowledge in expert systems is as a set of production rules, which are often coupled with a frame system that defines the objects that occur in the rules. In Section 8.2, we saw one example of an expert system rule, which was taken from the MYCIN system. Let's look at a few additional examples drawn from some other representative expert systems. All the rules we show are English versions of the actual rules that the systems use. Differences among these rules illustrate some of the important differences in the ways that expert systems operate.

Rl [McDermott, 1982; McDermott, 19841 (sometimes also called XCON) is a program that configures DEC VAX systems. Its rules look like this:

```
If: the most current active context is distributing
        massbus devices, and
    there is a single-port disk drive that has not been
        assigned to a massbus, and
    there are no unassigned dual-port disk drives, and
        the number of devices that each massbus should
        support is known, and
    there is a massbus that has been assigned at least
        one disk drive and that should support additional
        disk drives,
    and the type of cable needed to connect the disk drive
        to the previous device on the massbus is known
then: assign the disk drive to the massbus.
```

Notice that Rl's rules, unlike MYCIN's, contain no numeric measures of certainty. In the task domain with which Rl deals, it is possible to state exactly the correct thing to be done in each particular set of circumstances (although it may require a relatively complex set of antecedents to do so). One reason for this is that there exists a good deal of human expertise in this area. Another is that since Rl is doing a design task (in contrast to the diagnosis task performed by MYCIN), it is not necessary to consider all possible alternatives; one good one is enough. As a result, probabilistic information is not necessary in R1.

PROSPECTOR [Duda *et al.*. 1979; Hart *et al.*, 1978] is a program that provides advice on mineral exploration. Its rules look like this:

```
If: magnetite or pyrite in disseminated or veinlet form is present
then: (2, -4) there is favorable mineralization and texture
    for the propylitic stage.
```

In PROSPECTOR, each rule contains two confidence estimates. The first indicates the extent to which the presence of the evidence described in the condition part of the rule suggests the validity of the rule's conclusion. In the PROSPECTOR rule shown above, the number 2 indicates that the presence of the evidence is mildly encouraging. The second confidence estimate measures the extent to which the evidence is necessary to the validity of the conclusion, or stated another way, the extent to which the lack of the evidence indicates that the conclusion is not valid. In the example rule shown above, the number $-4$ indicates that the absence of the evidence is strongly discouraging for the conclusion.

DESIGN ADVISOR [Steele *et al.*, 1989] is a system that critiques chip designs. Its rules look like:

```
If: the sequential level count of ELEMENT is greater than 2,
    UNLESS the signal of ELEMENT is resetable
then: critique for poor resetability
DEFEAT: poor resetability of ELEMENT
due to: sequential level count of ELEMENT greater than 2
by: ELEMENT is directly resetable
```

The DESIGN ADVISOR gives advice to a chip designer, who can accept or reject the advice. If the advice is rejected, the system can exploit a justification-based truth maintenance system to revise its model of the circuit. The first rule shown here says that an element should be criticized for poor resetability if its sequential level count is greater than two, unless its signal is currently believed to be resetable. Resetability is a fairly common condition, so it is mentioned explicitly in this first rule. But there is also a much less common condition, called direct resetability. The DESIGN ADVISOR does not even bother to consider that condition unless it gets in trouble with its advice. At that point, it can exploit the second of the rules shown above. Specifically, if the chip designer rejects a critique about resetability and if that critique was based on a high level count, then the system will attempt to discover (possibly by asking the designer) whether the element is directly resetable. If it is, then the original rule is defeated and the conclusion withdrawn.

### *Reasoning with the Knowledge*

As these example rules have shown, expert systems exploit many of the representation and reasoning mechanisms that we have discussed. Because these programs are usually written primarily as rule-based systems, forward chaining, backward chaining, or some combination of the two, is usually used. For example, MYCIN used backward chaining to discover what organisms were present; then it used forward chaining to reason from the organisms to a treatment regime. Rl, on the other hand, used forward chaining. As the field of expert systems matures, more systems that exploit other kinds of reasoning mechanisms are being developed. The DESIGN ADVISOR is an example of such a system; in addition to exploiting rules, it makes extensive use of a justification-based truth maintenance system.

## 20.2 EXPERT SYSTEM SHELLS

Initially, each expert system that was built was created from scratch, usually in LISP. But, after several systems had been built this way, it became clear that these systems often had a lot in common. In particular, since the systems were constructed as a set of declarative representations (mostly rules) combined with an interpreter for those representations, it was possible to separate the interpreter from the domain-specific knowledge and thus to create a system that could be used to construct new expert systems by adding new knowledge corresponding to the new problem domain. The resulting interpreters are called *shells.* One influential example of such a shell is EMYCIN (for Empty MYCIN) [Buchanan and Shortliffe, 1984], which was derived from MYCIN.

There are now several commercially available shells that serve as the basis for many of the expert systems currently being built. These shells provide much greater flexibility in representing knowledge and in reasoning with it than MYCIN did. They typically support rules, frames, truth maintenance systems, and a variety of other reasoning mechanisms.

Early expert system shells provided mechanisms for knowledge representation, reasoning, and explanation. Later, tools for knowledge acquisition were added, as we see in Section 20.4. But as experience with using these systems to solve real world problems grew, it became clear that expert system shells needed to do something else as well. They needed to make it easy to integrate expert systems with other kinds of programs. Expert systems cannot operate in a vacuum, any more than their human counterparts can. They need access to corporate databases, and access to them needs to be controlled just as it does for other systems. They are often embedded within larger application programs that use primarily conventional programming techniques. So one of the important features that a shell must provide is an easy-to-use interface between an expert system that is written with the shell and a larger, probably more conventional, programming environment.

## 20.3 EXPLANATION

In order for an expert system to be an effective tool, people must be able to interact with it easily. To facilitate this interaction, the expert system must have the following two capabilities in addition to the ability to perform its underlying task:

- Explain its reasoning. In many of the domains in which expert systems operate, people will not accept results unless they have been convinced of the accuracy of the reasoning process that produced those results. This is particularly true, for example, in medicine, where a doctor must accept ultimate responsibility for a diagnosis, even if that diagnosis was arrived at with considerable help from a program. Thus it is important that the reasoning process used in such programs proceed in understandable steps and that enough meta-knowledge (knowledge about the reasoning process) be available so the explanations of those steps can be generated.
- Acquire new knowledge and modifications of old knowledge. Since expert systems derive their power from the richness of the knowledge bases they exploit, it is extremely important that those knowledge bases be as complete and as accurate as possible. But often there exists no standard codification of that knowledge; rather it exists only inside the heads of human experts. One way to get this knowledge into a program is through interaction with the human expert. Another way is to have the program learn expert behavior from raw data.

TEIRESIAS [Davis, 1982; Davis, 1977] was the first program to support explanation and knowledge acquisition. TEIRESIAS served as a front-end for the MYCIN expert system. A fragment of a TEIRESIAS-MYCIN conversation with a user (a doctor) is shown in Fig. 20.1. The program has asked for a piece of information that it needs in order to continue its reasoning. The doctor wants to know why the program wants the information, and later asks how the program arrived at a conclusion that it claimed it had reached.

An important premise underlying TEIRESIAS's approach to explanation is that the behavior of a program can be explained simply by referring to a trace of the program's execution. There are ways in which this assumption limits the kinds of explanations that can be produced, but it does minimize the overhead involved in generating each explanation. To understand how TEIRESIAS generates explanations of MYCIN's behavior, we need to know how that behavior is structured.

MYCIN attempts to solve its goal of recommending a therapy for a particular patient by first finding the cause of the patient's illness. It uses its production rules to reason backward from goals to clinical observations. To solve the top-level diagnostic goal, it looks for rules whose right sides suggest diseases. It then uses the left sides of those rules (the preconditions) to set up subgoals whose success would enable the rules to be invoked. These subgoals are again matched against rules, and their preconditions are used to set up additional subgoals. Whenever a precondition describes a specific piece of clinical evidence, MYCIN uses that evidence if it already has access to it. Otherwise, it asks the user to provide the information. In order that MYCIN's requests for information will appear coherent to the user, the actual goals that MYCIN sets up are often more general than they need be to satisfy the preconditions of an individual rule. For example, if a precondition specifies that the identity of an organism is *X*, MYCIN will set up the goal "infer identity." This approach also means that if another rule mentions the organism's identity, no further work will be required, since the identity will be known.

We can now return to the trace of TEIRESIAS-MYCIN's behavior shown in Fig. 20.1. The first question that the user asks is a "WHY" question, which is assumed to mean "Why do you need to know that?" Particularly for clinical tests that are either expensive or dangerous, it is important for the doctor to be convinced that the information is really needed before ordering the test. (Requests for sensitive or confidential information present similar difficulties.) Because MYCIN is reasoning backward, the question can easily be answered by examining the goal tree. Doing so provides two kinds of information:

- What higher-level question might the system be able to answer if it had the requested piece of information? (In this case, it could help determine the category of ORGANISM-1.)
- What other information does the system already have that makes it think that the requested piece of knowledge would help? (In this case, facts [2.1] to [2.4].)

```
Is the patient's illness with ORGANISM-1 a hospital-acquired infection?
** WHY
[i.e., WHY is it important to determine whether or not the
infection with ORGANISM-1 was acquired while the patient
was hospitalized?]
[2.0]   This will aid in determining the category of
        ORGANISM-1. It has already been established that
[2.1]   the gram stain of ORGANISM-1 is gramneg, and
[2.2]   the morphology of ORGANISM-1 is rod, and
[2.3]   the aerobicity of ORGANISM-1 is facultative
        Therefore, if
[2.4]   the infection with ORGANISM-1 was not acquired
        while the patient was hospitalized, then
there is strongly suggestive evidence (.8) that the
category of ORGANISM-1 is enterobacteriaceae
 [ALSO: there is weakly suggestive evidence (.1) that
 the identity of ORGANISM-1 is pseudomonas-aeruginosa]
 [RULE037]
** HOW [2.3]
[i.e., HOW was it established that the aerobicity of
 ORGANISM-1 is facultative?]
The following were used:
[3.1]   RULE027 indicated there is strongly suggestive
         evidence (.8) that the aerobicity of ORGANISM-1
         is facultative.
[3.2]   RULE033 indicated there is weakly suggestive
         evidence (.2) that the aerobicity of ORGANISM-1
         is not facultative.
Since this gave a cumulative CF of (.6) for facultative,
the evidence indicates that the aerobicity of ORGANISM-1
is facultative.
```

**Fig. 20.1** *A Portion of a Dialogue with TEIRESIAS*

When TEIRESIAS provides the answer to the first of these questions, the user may be satisfied or may want to follow the reasoning process back even further. The user can do that by asking additional "WHY" questions.

When TEIRESIAS provides the answer to the second of these questions and tells the user what it already believes, the user may want to know the basis for those beliefs. The user can ask this with a "HOW" question, which TEIRESIAS will interpret as "How did you know that?" This question also can be answered by looking at the goal tree and chaining backward from the stated fact to the evidence that allowed a rule that determined the fact to fire. Thus we see that by reasoning backward from its top-level goal and by keeping track of the entire tree that it traverses in the process, TEIRESIAS- MYCIN can do a fairly good job of justifying its reasoning to a human user. For more details of this process, as well as a discussion of some of its limitations, see Davis [1982].

The production system model is very general, and without some restrictions, it is hard to support all the kinds of explanations that a human might want. If we focus on a particular type of problem solving, we can ask more probing questions. For example, SALT [Marcus and McDermott, 1989] is a knowledge acquisition program used to build expert systems that design artifacts through a *propose-and-revise* strategy. SALT is capable of answering questions like WHY-NOT ("why didn't you assign value *x* to this parameter?") and WHAT-IF ("what would happen if you did?"). A human might ask these questions in order to locate incorrect or missing knowledge in the system as a precursor to correcting it. We now turn to ways in which a program such as SALT can support the process of building and refining knowledge.

## 20.4 KNOWLEDGE ACQUISITION

How are expert systems built? Typically, a knowledge engineer interviews a domain expert to elucidate expert knowledge, which is then translated into rules. After the initial system is built, it must be iteratively refined until it approximates expert-level performance. This process is expensive and time-consuming, so it is worthwhile to look for more automatic ways of constructing expert knowledge bases. While no totally automatic knowledge acquisition systems yet exist, there are many programs that interact with domain experts to extract expert knowledge efficiently. These programs provide support for the following activities:

- Entering knowledge
- Maintaining knowledge base consistency
- Ensuring knowledge base completeness

The most useful knowledge acquisition programs are those that are restricted to a particular problem-solving paradigm, e.g., diagnosis or design. It is important to be able to enumerate the roles that knowledge can play in the problem-solving process. For example, if the paradigm is diagnosis, then the program can structure its knowledge base around symptoms, hypotheses, and causes. It can identify symptoms for which the expert has not yet provided causes. Since one symptom may have multiple causes, the program can ask for knowledge about how to decide when one hypothesis is better than another. If we move to another type of problem-solving, say designing artifacts, then these acquisition strategies no longer apply, and we must look for other ways of profitably interacting with an expert. We now examine two knowledge acquisition systems in detail.

MOLE [Eshelman, 1988] is a knowledge acquisition system for heuristic classification problems, such as diagnosing diseases. In particular, it is used in conjunction with the *cover-and-differentiate* problem-solving method. An expert system produced by MOLE accepts input data, comes up with a set of candidate explanations or classifications that cover (or explain) the data, then uses differentiating knowledge to determine which one is best. The process is iterative, since explanations must themselves be justified, until ultimate causes are ascertained.

MOLE interacts with a domain expert to produce a knowledge base that a system called MOLE-p (for MOLE-performance) uses to solve problems. The aequisition proceeds through several steps:

1. Initial knowledge base construction. MOLE asks the expert to list common symptoms or complaints that might require diagnosis. For each symptom, MOLE prompts for a list of possible explanations. MOLE then iteratively seeks out higher-level explanations until it comes up with a set of ultimate causes. During this process, MOLE builds an influence network similar to the belief networks we saw in Chapter 8.

   Whenever an event has multiple explanations, MOLE tries to determine the conditions under which one explanation is correct. The expert provides *covering* knowledge, that is, the knowledge that a hypothesized event might be the cause of a certain symptom. MOLE then tries to infer *anticipatory* knowledge, which says that if the hypothesized event does occur, then the symptom will definitely appear. This knowledge allows the system to rule out certain hypotheses on the basis that specific symptoms are absent.
2. Refinement of the knowledge base. MOLE now tries to identify the weaknesses of the knowledge base. One approach is to find holes and prompt the expert to fill them. It is difficult, in general, to know whether a knowledge base is complete, so instead MOLE lets the expert watch MOLE-p solving sample

problems. Whenever MOLE-p makes an incorrect diagnosis, the expert adds new knowledge. There are several ways in which MOLE-p can reach the wrong conclusion. It may incorrectly reject a hypothesis because it does not feel that the hypothesis is needed to explain any symptom. It may advance a hypothesis because it is needed to explain some otherwise inexplicable hypothesis. Or it may lack differentiating knowledge for choosing between alternative hypotheses.

For example, suppose we have a patient with symptoms A and B. Further suppose that symptom A could be caused by events X and Y, and that symptom B can be caused by Y and Z. MOLE-p might conclude Y, since it explains both A and B. If the expert indicates that this decision was incorrect, then MOLE will ask what evidence should be used to prefer X and/or Z over Y.

MOLE has been used to build systems that diagnose problems with car engines, problems in steel-rolling mills, and inefficiencies in coal-burning power plants. For MOLE to be applicable, however, it must be possible to preenumerate solutions or classifications. It must also be practical to encode the knowledge in terms of covering and differentiating.

But suppose our task is to design an artifact, for example, an elevator system. It is no longer possible to preenumerate all solutions. Instead, we must assign values to a large number of parameters, such as the width of the platform, the type of door, the cable weight, and the cable strength. These parameters must be consistent with each other, and they must result in a design that satisfies external constraints imposed by cost factors, the type of building involved, and expected payloads.

One problem-solving method useful for design tasks is called *propose-and-revise*. Propose-and-revise systems build up solutions incrementally. First, the system proposes an extension to the current design. Then it checks whether the extension violates any global or local constraints. Constraint violations are then fixed, and the process repeats. It turns out that domain experts are good at listing overall design constraints and at providing local constraints on individual parameters, but not so good at explaining how to arrive at global solutions. The SALT program [Marcus and McDermott, 1989] provides mechanisms for elucidating this knowledge from the expert.

Like MOLE, SALT builds a dependency network as it converses with the expert. Each node stands for a value of a parameter that must be acquired or generated. There are three kinds of links: *contributes-to, constrains,* and *suggests-revision-of.* Associated with the first type of link are procedures that allow SALT to generate a value for one parameter based on the value of another. The second type of link, *constrains,* rules out certain parameter values. The third link, *suggests-revision-of,* points to ways in which a constraint violation can be fixed. SALT uses the following heuristics to guide the acquisition process:

1. Every noninput node in the network needs at least one *contributes-to* link coming into it. If links are missing, the expert is prompted to fill them in.
2. No *contributes-to* loops are allowed in the network. Without a value for at least one parameter in the loop, it is impossible to compute values for any parameter in that loop. If a loop exists, SALT tries to transform one of the *contributes-to* links into a *constrains* link.
3. Constraining links should have *suggests-revision-of* links associated with them. These include *constrains* links that are created when dependency loops are broken.

Control knowledge is also important. It is critical that the system propose extensions and revisions that lead toward a design solution. SALT allows the expert to rate revisions in terms of how much trouble they tend to produce.

SALT compiles its dependency network into a set of production rules. As with MOLE, an expert can watch the production system solve problems and can override the system's decision. At that point, the knowledge base can be changed or the override can be logged for future inspection.

The process of interviewing a human expert to extract expertise presents a number of difficulties, regardless of whether the interview is conducted by a human or by a machine. Experts are surprisingly inarticulate when it comes to how they solve problems. They do not seem to have access to the low-level details of what they do and are especially inadequate suppliers of any type of statistical information. There is, therefore, a great deal of interest in building systems that automatically induce their own rules by looking at sample problems and solutions. With inductive techniques, an expert needs only to provide the conceptual framework for a problem and a set of useful examples.

For example, consider a bank's problem in deciding whether to approve a loan. One approach to automating this task is to interview loan officers in an attempt to extract their domain knowledge. Another approach is to inspect the record of loans the bank has made in the past and then try to generate automatically rules that will maximize the number of good loans and minimize the number of bad ones in the future.

META-DENDRAL [Mitchell, 1978] was the first program to use learning techniques to construct rules for an expert system automatically. It built rules to be used by DENDRAL, whose job was to determine the structure of complex chemical compounds. META-DENDRAL was able to induce its rules based on a set of mass spectrometry data; it was then able to identify molecular structures with very high accuracy. META-DENDRAL used the version space learning algorithm, which we discussed in Chapter 17. Another popular method for automatically constructing expert systems is the induction of decision trees, data structures we described in Section 17.5.3. Decision tree expert systems have been built for assessing consumer credit applications, analyzing hypothyroid conditions, and diagnosing soybean diseases, among many other applications.

Statistical techniques, such as multivariate analysis, provide an alternative approach to building expert-level systems. Unfortunately, statistical methods do not produce concise rules that humans can understand. Therefore it is difficult for them to explain their decisions.

For highly structured problems that require deep causal chains of reasoning, learning techniques are presently inadequate. There is, however, a great deal of research activity in this area, as we saw in Chapter 17.

## SUMMARY

Since the mid-1960s, when work began on the earliest of what are now called expert systems, much progress has been made in the construction of such programs. Experience gained in these efforts suggests the following conclusions:

- These systems derive their power from a great deal of domain-specific knowledge, rather than from a single powerful technique.
- In successful systems, the required knowledge is about a particular area and is well defined. This contrasts with the kind of broad, hard-to-define knowledge that we call common sense. It is easier to build expert systems than ones with common sense.
- An expert system is usually built with the aid of one or more experts, who must be willing to spend a great deal of effort transferring their expertise to the system.
- Transfer of knowledge takes place gradually through many interactions between the expert and the system. The expert will never get the knowledge right or complete the first time.
- The amount of knowledge that is required depends on the task. It may range from forty rules to thousands.
- The choice of control structure for a particular system depends on specific characteristics of the system.
- It is possible to extract the nondomain-specific parts from existing expert systems and use them as tools for building new systems in new domains.