most such underconstrained problems. On the other hand, an overconstrained problem has many clauses relative to the number of variables and is likely to have no solutions.

To go beyond these basic intuitions, we must define exactly how random sentences are generated. The notation $CNF_k(m, n)$ denotes a $k$-CNF sentence with $m$ clauses and $n$ symbols, where the clauses are chosen uniformly, independently, and without replacement from among all clauses with $k$ different literals, which are positive or negative at random. (A symbol may not appear twice in a clause, nor may a clause appear twice in a sentence.)

Given a source of random sentences, we can measure the probability of satisfiability. Figure 7.19(a) plots the probability for $CNF_3(m, 50)$, that is, sentences with 50 variables and 3 literals per clause, as a function of the clause/symbol ratio, $m/n$. As we expect, for small $m/n$ the probability of satisfiability is close to 1, and at large $m/n$ the probability is close to 0. The probability drops fairly sharply around $m/n = 4.3$. Empirically, we find that the “cliff” stays in roughly the same place (for $k = 3$) and gets sharper and sharper as $n$ increases. Theoretically, the satisfiability threshold conjecture says that for every $k > 3$, there is a threshold ratio $r_k$ such that, as $n$ goes to infinity, the probability that $CNF(n, r_n)$ is satisfiable becomes 1 for all values of $r$ below the threshold, and 0 for all values above. The conjecture remains unproven.

Figure 7.19  (a) Graph showing the probability that a random 3-CNF sentence with $n = 50$ symbols is satisfiable, as a function of the clause/symbol ratio $m/n$. (b) Graph of the median run time (measured in number of recursive calls to DPLL, a good proxy) on random 3-CNF sentences. The most difficult problems have a clause/symbol ratio of about 4.3.

Now that we have a good idea where the satisfiable and unsatisfiable problems are, the next question is, where are the hard problems? It turns out that they are also often at the threshold value. Figure 7.19(b) shows that 50-symbol problems at the threshold value of 4.3 are about 20 times more difficult to solve than those at a ratio of 3.3. The underconstrained problems are easiest to solve (because it is so easy to guess a solution); the overconstrained problems are not as easy as the underconstrained, but still are much easier than the ones right at the threshold.
7.7 AGENTS BASED ON PROPOSITIONAL LOGIC

In this section, we bring together what we have learned so far in order to construct wumpus world agents that use propositional logic. The first step is to enable the agent to deduce, to the extent possible, the state of the world given its percept history. This requires writing down a complete logical model of the effects of actions. We also show how the agent can keep track of the world efficiently without going back into the percept history for each inference. Finally, we show how the agent can use logical inference to construct plans that are guaranteed to achieve its goals.

7.7.1 The current state of the world

As stated at the beginning of the chapter, a logical agent operates by deducing what to do from a knowledge base of sentences about the world. The knowledge base is composed of axioms—general knowledge about how the world works—and percept sentences obtained from the agent’s experience in a particular world. In this section, we focus on the problem of deducing the current state of the wumpus world—where am I, is that square safe, and so on.

We began collecting axioms in Section 7.4.3. The agent knows that the starting square contains no pit (–P1,1) and no wumpus (–W1,1). Furthermore, for each square, it knows that the square is breezy if and only if a neighboring square has a pit; and a square is smelly if and only if a neighboring square has a wumpus. Thus, we include a large collection of sentences of the following form:

\[ S_{1,1} \leftrightarrow (P_{1,2} \lor P_{2,1}) \]
\[ S_{1,4} \leftrightarrow (W_{1,2} \lor W_{2,1}) \]

The agent also knows that there is exactly one wumpus. This is expressed in two parts. First, we have to say that there is at least one wumpus:

\[ W_{1,1} \lor W_{1,2} \lor \cdots \lor W_{4,3} \lor W_{4,4} \]

Then, we have to say that there is at most one wumpus. For each pair of locations, we add a sentence saying that at least one of them must be wumpus-free:

\[ W_{1,1} \lor W_{1,2} \]
\[ W_{1,1} \lor W_{1,2} \]
\[ W_{1,1} \lor W_{1,2} \]
\[ W_{1,1} \lor W_{1,2} \]
\[ W_{1,1} \lor W_{1,2} \]

So far, so good. Now let’s consider the agent’s percepts. If there is currently a stench, one might suppose that a proposition Stench should be added to the knowledge base. This is not quite right, however: if there was no stench at the previous time step, then –Stench would already be asserted, and the new assertion would simply result in a contradiction. The problem is solved when we realize that a percept asserts something only about the current time. Thus, if the time step (as supplied to MAKE-PERCEPT-SENTENCE in Figure 7.1) is 4, then we add