sharp peak in the run time of their SAT solver. Cook and Mitchell (1997) provide an excellent summary of the early literature on the problem.

The current state of theoretical understanding is summarized by Achlioptas (2009). The satisfiability threshold conjecture states that, for each $k$, there is a sharp satisfiability threshold $r_k$, such that as the number of variables $n \to \infty$, instances below the threshold are satisfiable with probability 1, while those above the threshold are unsatisfiable with probability 1. The conjecture was not quite proved by Friedgut (1999): a sharp threshold exists but its location might depend on $n$ even as $n \to \infty$. Despite significant progress in asymptotic analysis of the threshold location for large $k$ (Achlioptas and Peres, 2004; Achlioptas et al., 2007), all that can be proved for $k = 3$ is that it lies in the range $[3.52, 4.51]$. Current theory suggests that a peak in the run time of a SAT solver is not necessarily related to the satisfiability threshold, but instead to a phase transition in the solution distribution and structure of SAT instances. Empirical results due to Coarfa et al. (2003) support this view. In fact, algorithms such as survey propagation (Parisi and Zecchina, 2002; Maneva et al., 2007) take advantage of special properties of random SAT instances near the satisfiability threshold and greatly outperform general SAT solvers on such instances.

The best sources for information on satisfiability, both theoretical and practical, are the Handbook of Satisfiability (Biere et al., 2009) and the regular International Conferences on Theory and Applications of Satisfiability Testing, known as SAT.

The idea of building agents with propositional logic can be traced back to the seminal paper of McCulloch and Pitts (1943), which initiated the field of neural networks. Contrary to popular supposition, the paper was concerned with the implementation of a Boolean circuit-based agent design in the brain. Circuit-based agents, which perform computation by propagating signals in hardware circuits rather than running algorithms in general-purpose computers, have received little attention in AI, however. The most notable exception is the work of Stan Rosenschein (Rosenschein, 1985; Kaelbling and Rosenschein, 1990), who developed ways to compile circuit-based agents from declarative descriptions of the task environment. (Rosenschein’s approach is described at some length in the second edition of this book.) The work of Rod Brooks (1986, 1989) demonstrates the effectiveness of circuit-based designs for controlling robots—a topic we take up in Chapter 25. Brooks (1991) argues that circuit-based designs are all that is needed for AI—that representation and reasoning are cumbersome, expensive, and unnecessary. In our view, neither approach is sufficient by itself. Williams et al. (2003) show how a hybrid agent design not too different from our wumpus agent has been used to control NASA spacecraft, planning sequences of actions and diagnosing and recovering from faults.

The general problem of keeping track of a partially observable environment was introduced for state-based representations in Chapter 4. Its instantiation for propositional representations was studied by Amir and Russell (2003), who identified several classes of environments that admit efficient state-estimation algorithms and showed that for several other classes the problem is intractable. The temporal-projection problem, which involves determining what propositions hold true after an action sequence is executed, can be seen as a special case of state estimation with empty percepts. Many authors have studied this problem because of its importance in planning; some important hardness results were established by
The idea of representing a belief state with propositions can be traced to Wittgenstein (1922). Logical state estimation, of course, requires a logical representation of the effects of actions—a key problem in AI since the late 1950s. The dominant proposal has been the situation calculus formalism (McCarthy, 1963), which is couched within first-order logic. We discuss situation calculus, and various extensions and alternatives, in Chapters 10 and 12. The approach taken in this chapter—using temporal indices on propositional variables—is more restrictive but has the benefit of simplicity. The general approach embodied in the SATPLAN algorithm was proposed by Kautz and Selman (1992). Later generations of SATPLAN were able to take advantage of the advances in SAT solvers, described earlier, and remain among the most effective ways of solving difficult problems (Kautz, 2006).

The frame problem was first recognized by McCarthy and Hayes (1969). Many researchers considered the problem unsolvable within first-order logic, and it spurred a great deal of research into nonmonotonic logics. Philosophers from Dreyfus (1972) to Crockett (1994) have cited the frame problem as one symptom of the inevitable failure of the entire AI enterprise. The solution of the frame problem with successor-state axioms is due to Ray Reiter (1991). Thielischer (1999) identifies the inferential frame problem as a separate idea and provides a solution. In retrospect, one can see that Rosenschein's (1985) agents were using circuits that implemented successor-state axioms, but Rosenschein did not notice that the frame problem was thereby largely solved. Foo (2001) explains why the discrete-event control theory models typically used by engineers do not have to explicitly deal with the frame problem: because they are dealing with prediction and control, not with explanation and reasoning about counterfactual situations.

Modern propositional solvers have wide applicability in industrial applications. The application of propositional inference in the synthesis of computer hardware is now a standard technique having many large-scale deployments (Nowick et al., 1993). The SATMC satisfiability checker was used to detect a previously unknown vulnerability in a Web browser user sign-on protocol (Armando et al., 2008).

The wumpus world was invented by Gregory Yoh (1975). Ironically, Yoh developed it because he was bored with games played on a rectangular grid: the topology of his original wumpus world was a dodecahedron, and we put it back in the boring old grid. Michael Genesereth was the first to suggest that the wumpus world be used as an agent testbed.

**Exercises**

7.1 Suppose the agent has progressed to the point shown in Figure 7.4(a), page 239, having perceived nothing in [1,1], a breeze in [2,1], and a stench in [1,2], and is now concerned with the contents of [1,3], [2,2], and [3,1]. Each of these can contain a pit, and at most one can contain a wumpus. Following the example of Figure 7.5, construct the set of possible worlds. (You should find 32 of them.) Mark the worlds in which the KB is true and those in which