2. Present the whole sentence to a SAT solver. If the solver finds a satisfying model, then the goal is achievable; if the sentence is unsatisfiable, then the planning problem is impossible.

3. Assuming a model is found, extract from the model those variables that represent actions and are assigned true. Together they represent a plan to achieve the goals.

A propositional planning procedure, SATPLAN, is shown in Figure 7.22. It implements the basic idea just given, with one twist. Because the agent does not know how many steps it will take to reach the goal, the algorithm tries each possible number of steps $t$, up to some maximum conceivable plan length $T_{\text{max}}$. In this way, it is guaranteed to find the shortest plan if one exists. Because of the way SATPLAN searches for a solution, this approach cannot be used in a partially observable environment; SATPLAN would just set the unobservable variables to the values it needs to create a solution.

```plaintext
function SATPLAN("init", transition, goal, $T_{\text{max}}$) returns solution or failure
  inputs: in it, transition, goal, constitute a description of the problem
  $T_{\text{max}}$, an upper limit for plan length
  for $t = 0$ to $T$, do
    $cnf = \text{translate-to-SAT}(\text{init}, \text{transition}, \text{goal}, t)$
    $model = \text{SAT-SOLVE}(cnf)$
    if $model$ is not null then
      return extract-solution($model$)
  return failure
```

Figure 7.22 The SATPLAN algorithm. The planning problem is translated into a CNF sentence in which the goal is asserted to hold at a fixed time step $t$ and axioms are included for each time step up to $t$. If the satisfiability algorithm finds a model, then a plan is extracted by looking at those proposition symbols that refer to actions and are assigned true in the model. If no model exists, then the process is repeated with the goal moved one step later.

The key step in using SATPLAN is the construction of the knowledge base. It might seem, on casual inspection, that the wumpus world axioms in Section 7.7.1 suffice for steps 1(a) and 1(b) above. There is, however, a significant difference between the requirements for entailment (as tested by ASK) and those for satisfiability. Consider, for example, the agent’s location, initially $[1,1]$, and suppose the agent’s unambitious goal is to be in $[2,1]$ at time 1. The initial knowledge base contains $L_{1,1}^0$ and the goal is $L_{2,1}^1$. Using ASK, we can prove $L_{2,1}^1$ if $\text{Forward}$ is asserted, and, reassuringly, we cannot prove $L_{2,1}^1$ if, say, $\text{Shoot}$ is asserted instead. Now, SATPLAN will find the plan $[\text{Forward}]$; so far, so good. Unfortunately, SATPLAN also finds the plan $[\text{Shoot}]$. How could this be? To find out, we inspect the model that SATPLAN constructs: it includes the assignment $L_{2,1}^0$, that is, the agent can be in $[2,1]$ at time 1 by being there at time 0 and shooting. One might ask, "Didn't we say the agent is in $[1,1]$ at time 0?" Yes, we did, but we didn't tell the agent that it can't be in two places at once! For entailment, $L_{2,1}^0$ is unknown and cannot, therefore, be used in a proof; for satisfiability,
Section 7.7. Agents Based on Propositional Logic

on the other hand, $L_2$ is unknown and can, therefore, be set to whatever value helps to make the goal true. For this reason, SATPLAN is a debugging tool for knowledge bases because it reveals places where knowledge is missing. In this particular case, we can fix the knowledge base by asserting that, at each time step, the agent is in exactly one location, using a collection of sentences similar to those used to assert the existence of exactly one wumpus. Alternatively, we can assert for all locations other than $[1,1]$; the successor-state axiom for location takes care of subsequent time steps. The same fixes also work to make sure the agent has only one orientation.

SATPLAN has more surprises in store, however. The first is that it finds models with impossible actions, such as shooting with no arrow. To understand why, we need to look more carefully at what the successor-state axioms (such as Equation (7.3)) say about actions whose preconditions are not satisfied. The axioms do predict correctly that nothing will happen when such an action is executed (see Exercise 10.14), but they do not say that the action cannot be executed! To avoid generating plans with illegal actions, we must add precondition axioms stating that an action occurrence requires the preconditions to be satisfied. For example, we need to say, for each time $t$, that

\[ \text{Shoot} = \text{HaveArrow} \]

This ensures that if a plan selects the Shoot action at any time, it must be the case that the agent has an arrow at that time.

SATPLAN’s second surprise is the creation of plans with multiple simultaneous actions. For example, it may come up with a model in which both Forward and Shoot are true, which is not allowed. To eliminate this problem, we introduce action exclusion axioms: for every pair of actions $A_i$ and $A_j$, we add the axiom

\[ \lor \]

It might be pointed out that walking forward and shooting at the same time is not so hard to do, whereas, say, shooting and grabbing at the same time is rather impractical. By imposing action exclusion axioms only on pairs of actions that really do interfere with each other, we can allow for plans that include multiple simultaneous actions—and because SATPLAN finds the shortest legal plan, we can be sure that it will take advantage of this capability.

To summarize, SATPLAN finds models for a sentence containing the initial state, the goal, the successor-state axioms, the precondition axioms, and the action exclusion axioms. It can be shown that this collection of axioms is sufficient, in the sense that there are no longer any spurious “solutions.” Any model satisfying the propositional sentence will be a valid plan for the original problem. Modern SAT-solving technology makes the approach quite practical. For example, a DPLL-style solver has no difficulty in generating the 11-step solution for the wumpus world instance shown in Figure 7.2.

This section has described a declarative approach to agent construction: the agent works by a combination of asserting sentences in the knowledge base and performing logical inference. This approach has some weaknesses hidden in phrases such as “for each time $t$” and

---

Notice that the addition of precondition axioms means that we need not include preconditions for actions in the successor-state axioms.