effect if there is a wall in front of the agent. The action *Grab* can be used to pick up a object that is in the same square as the agent. The action *Shoot* can be used to fire a arrow in a straight line in the direction the agent is facing. The arrow continues unti either hits (and hence kills) the wumpus or hits a wall. The agent only has one arr so only the first *Shoot* action has any effect.

◇ **Sensors**: The agent has five sensors, each of which gives a single bit of information

- In the square containing the wumpus and in the directly (not diagonally) adjac squares the agent will perceive a stench.
- In the squares directly adjacent to a pit, the agent will perceive a breeze.
- In the square where the gold is, the agent will perceive a glitter.
- When an agent walks into a wall, it will perceive a bump.
- When the wumpus is killed, it emits a woeful scream that can be perceived a where in the cave.

The percepts will be given to the agent in the form of a list of five symbols; for examp if there is a stench and a breeze, but no glitter, bump, or scream, the agent will rece the percept $[Stench, Breeze, None, None, None]$.

Exercise 7.1 asks you to define the wumpus environment along the various dimensions gi in Chapter 2. The principal difficulty for the agent is its initial ignorance of the configurat of the environment; overcoming this ignorance seems to require logical reasoning. In m instances of the wumpus world, it is possible for the agent to retrieve the gold safely. Oc sionally, the agent must choose between going home empty-handed and risking death to f the gold. About 21% of the environments are utterly unfair, because the gold is in a pit o surrounded by pits.

Let us watch a knowledge-based wumpus agent exploring the environment shown i Figure 7.2. The agent's initial knowledge base contains the rules of the environment, as lis

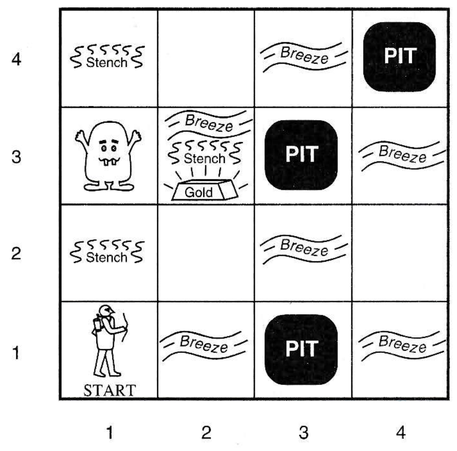

Figure 7.2 A typical wumpus world. The agent is in the bottom left corner.

previously; in particular, it knows that it is in [1,1] and that [1,1] is a safe square. We will see how its knowledge evolves as new percepts arrive and actions are taken.

The first percept is [*None*, *None*, *None*, *None*, *None*], from which the agent can conclude that its neighboring squares are safe. Figure 7.3(a) shows the agent's state of knowledge at this point. We list (some of) the sentences in the knowledge base using letters such as B (breezy) and OK (safe, neither pit nor wumpus) marked in the appropriate squares. Figure 7.2, on the other hand, depicts the world itself.

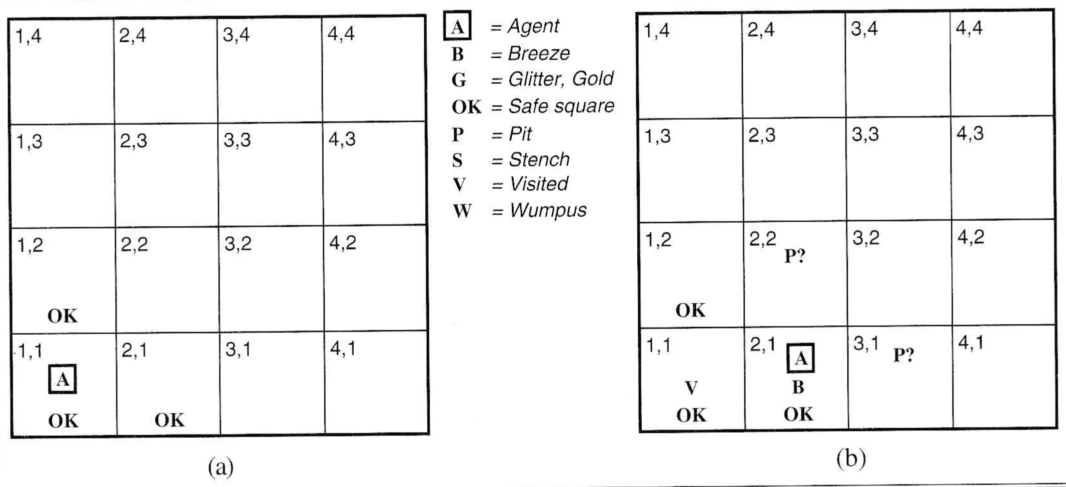

Figure 7.3 The first step taken by the agent in the wumpus world. (a) The initial situation, after percept [*None*, *None*, *None*, *None*, *None*]. (b) After one move, with percept [*None*, *Breeze*, *None*, *None*, *None*].

From the fact that there was no stench or breeze in [1,1], the agent can infer that [1,2] and [2,1] are free of dangers. They are marked with an OK to indicate this. A cautious agent will move only into a square that it knows is OK. Let us suppose the agent decides to move forward to [2,1], giving the scene in Figure 7.3(b).

The agent detects a breeze in [2,1], so there must be a pit in a neighboring square. The pit cannot be in [1,1], by the rules of the game, so there must be a pit in [2,2] or [3,1] or both. The notation $P?$ in Figure 7.3(b) indicates a possible pit in those squares. At this point, there is only one known square that is OK and has not been visited yet. So the prudent agent will turn around, go back to [1,1], and then proceed to [1,2].

The new percept in [1,2] is [*Stench*, *None*, *None*, *None*, *None*], resulting in the state of knowledge shown in Figure 7.4(a). The stench in [1,2] means that there must be a wumpus nearby. But the wumpus cannot be in [1,1], by the rules of the game, and it cannot be in [2,2] (or the agent would have detected a stench when it was in [2,1]). Therefore, the agent can infer that the wumpus is in [1,3]. The notation $W!$ indicates this. Moreover, the lack of a *Breeze* in [1,2] implies that there is no pit in [2,2]. Yet we already inferred that there must be a pit in either [2,2] or [3,1], so this means it must be in [3,1]. This is a fairly difficult inference, because it combines knowledge gained at different times in different places and

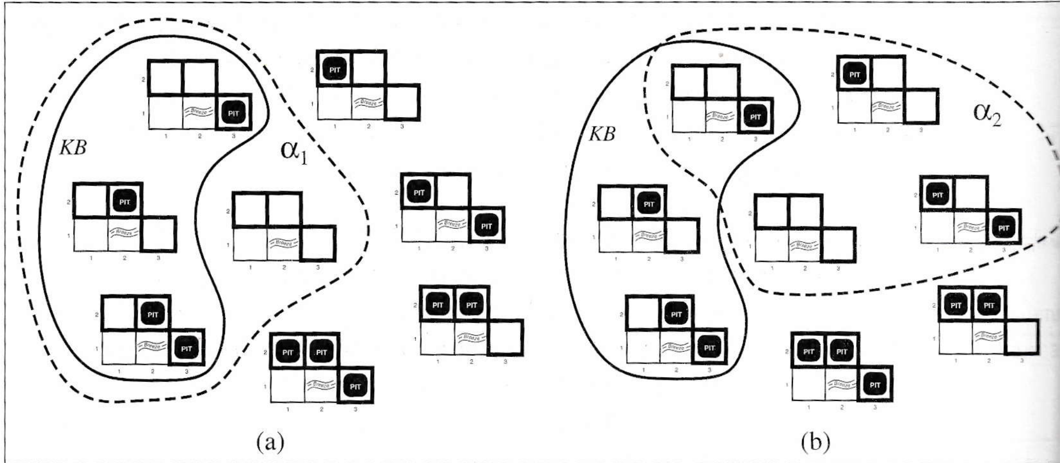

Figure 7.5 Possible models for the presence of pits in squares [1,2], [2,2], and [3,1], given observations of nothing in [1,1] and a breeze in [2,1]. (a) Models of the knowledge base and α_1 (no pit in [1,2]). (b) Models of the knowledge base and α_2 (no pit in [2,2]).

agent is interested (among other things) in whether the adjacent squares [1,2], [2,2], and [3, contain pits. Each of the three squares might or might not contain a pit, so (for the purpos of this example) there are $2^3 = 8$ possible models. These are shown in Figure 7.5.[3]

The KB is false in models that contradict what the agent knows—for example, the K is false in any model in which [1,2] contains a pit, because there is no breeze in [1,1]. The are in fact just three models in which the KB is true, and these are shown as a subset of t models in Figure 7.5. Now let us consider two possible conclusions:

α_1 = "There is no pit in [1,2]."
α_2 = "There is no pit in [2,2]."

We have marked the models of α_1 and α_2 in Figures 7.5(a) and 7.5(b) respectively. inspection, we see the following:

in every model in which KB is true, α_1 is also true.

Hence, $KB \models \alpha_1$: there is no pit in [1,2]. We can also see that

in some models in which KB is true, α_2 is false.

Hence, $KB \not\models \alpha_2$: the agent *cannot* conclude that there is no pit in [2,2]. (Nor can it concl that there *is* a pit in [2,2].)[4]

The preceding example not only illustrates entailment, but also shows how the def tion of entailment can be applied to derive conclusions—that is, to carry out **logical inf ence**. The inference algorithm illustrated in Figure 7.5 is called **model checking**, becaus enumerates all possible models to check that α is true in all models in which KB is true.

LOGICAL INFERENCE
MODEL CHECKING

[3] Although the figure shows the models as partial wumpus worlds, they are really nothing more than assignn of *true* and *false* to the sentences "there is a pit in [1,2]" etc. Models, in the mathematical sense, do not ne have 'orrible 'airy wumpuses in them.

[4] The agent can calculate the *probability* that there is a pit in [2,2]; Chapter 13 shows how.

A simple knowledge base

Now that we have defined the semantics for propositional logic, we can construct a knowledg base for the wumpus world. For simplicity, we will deal only with pits; the wumpus itse is left as an exercise. We will provide enough knowledge to carry out the inference that w done informally in Section 7.3.

First, we need to choose our vocabulary of proposition symbols. For each i, j:

- Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
- Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

The knowledge base includes the following sentences, each one labeled for convenience:

- There is no pit in [1,1]:

$$R_1: \quad \neg P_{1,1} .$$

- A square is breezy if and only if there is a pit in a neighboring square. This has to stated for each square; for now, we include just the relevant squares:

$$R_2: \quad B_{1,1} \quad \Leftrightarrow \quad (P_{1,2} \lor P_{2,1}) .$$
$$R_3: \quad B_{2,1} \quad \Leftrightarrow \quad (P_{1,1} \lor P_{2,2} \lor P_{3,1}) .$$

- The preceding sentences are true in all wumpus worlds. Now we include the bree percepts for the first two squares visited in the specific world the agent is in, leading u to the situation in Figure 7.3(b).

$$R_4: \quad \neg B_{1,1} .$$
$$R_5: \quad B_{2,1} .$$

The knowledge base, then, consists of sentences R_1 through R_5. It can also be considered a single sentence—the conjunction $R_1 \land R_2 \land R_3 \land R_4 \land R_5$—because it asserts that all individual sentences are true.

Inference

Recall that the aim of logical inference is to decide whether $KB \models \alpha$ for some sentence For example, is $P_{2,2}$ entailed? Our first algorithm for inference will be a direct implemen tion of the definition of entailment: enumerate the models, and check that α is true in ev model in which KB is true. For propositional logic, models are assignments of *true* or *fa* to every proposition symbol. Returning to our wumpus-world example, the relevant propo tion symbols are $B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,1}$, $P_{2,2}$, and $P_{3,1}$. With seven symbols, there $2^7 = 128$ possible models; in three of these, KB is true (Figure 7.9). In those three mode $\neg P_{1,2}$ is true, hence there is no pit in [1,2]. On the other hand, $P_{2,2}$ is true in two of the th models and false in one, so we cannot yet tell whether there is a pit in [2,2].

Figure 7.9 reproduces in a more precise form the reasoning illustrated in Figure 7.5 general algorithm for deciding entailment in propositional logic is shown in Figure 7.10. L the BACKTRACKING-SEARCH algorithm on page 76, TT-ENTAILS? performs a recurs enumeration of a finite space of assignments to variables. The algorithm is **sound**, becaus

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	*KB*
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Figure 7.9 A truth table constructed for the knowledge base given in the text. *KB* is true if R_1 through R_5 are true, which occurs in just 3 of the 128 rows. In all 3 rows, $P_{1,2}$ is false, so there is no pit in [1,2]. On the other hand, there might (or might not) be a pit in [2,2].

```
function TT-ENTAILS?($KB, \alpha$) returns true or false
  inputs: $KB$, the knowledge base, a sentence in propositional logic
          $\alpha$, the query, a sentence in propositional logic

  symbols ← a list of the proposition symbols in $KB$ and $\alpha$
  return TT-CHECK-ALL($KB, \alpha$, symbols, [ ])
```

```
function TT-CHECK-ALL($KB, \alpha$, symbols, model) returns true or false
  if EMPTY?(symbols) then
     if PL-TRUE?($KB$, model) then return PL-TRUE?($\alpha$, model)
     else return true
  else do
     $P$ ← FIRST(symbols); rest ← REST(symbols)
     return TT-CHECK-ALL($KB, \alpha$, rest, EXTEND($P$, true, model)) and
               TT-CHECK-ALL($KB, \alpha$, rest, EXTEND($P$, false, model))
```

Figure 7.10 A truth-table enumeration algorithm for deciding propositional entailment. TT stands for truth table. PL-TRUE? returns true if a sentence holds within a model. The variable *model* represents a partial model—an assignment to only some of the variables. The function call EXTEND(*P, true, model*) returns a new partial model in which P has the value *true*.

implements directly the definition of entailment, and **complete**, because it works for any KB and α and always terminates—there are only finitely many models to examine.

Of course, "finitely many" is not always the same as "few." If KB and α contain n symbols in all, then there are 2^n models. Thus, the time complexity of the algorithm is $O(2^n)$. (The space complexity is only $O(n)$ because the enumeration is depth-first.) Later in this