

7.2 The Wumpus World

In this section we describe an environment in which knowledge-based agents can show their worth. The **wumpus world** is a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the terrible wumpus, a beast that eats anyone who enters its room. The wumpus can be shot by an agent, but the agent has only one arrow. Some rooms contain bottomless pits that will trap anyone who wanders into these rooms (except for the wumpus, which is too big to fall in). The only redeeming feature of this bleak environment is the possibility of finding a heap of gold. Although the wumpus world is rather tame by modern computer game standards, it illustrates some important points about intelligence.

Wumpus world

A sample wumpus world is shown in [Figure 7.2](#). The precise definition of the task environment is given, as suggested in [Section 2.3](#), by the PEAS description:

- **PERFORMANCE MEASURE:** +1000 for climbing out of the cave with the gold, −1000 for falling into a pit or being eaten by the wumpus, −1 for each action taken, and −10 for using up the arrow. The game ends either when the agent dies or when the agent climbs out of the cave.
- **ENVIRONMENT:** A 4×4 grid of rooms, with walls surrounding the grid. The agent always starts in the square labeled [1,1], facing to the east. The locations of the gold and the wumpus are chosen randomly, with a uniform distribution, from the squares other than the start square. In addition, each square other than the start can be a pit, with probability 0.2.
- **ACTUATORS:** The agent can move *Forward*, *TurnLeft* by 90° , or *TurnRight* by 90° . The agent dies a miserable death if it enters a square containing a pit or a live wumpus. (It is safe, albeit smelly, to enter a square with a dead wumpus.) If an agent tries to move forward and bumps into a wall, then the agent does not move. The action *Grab* can be

used to pick up the gold if it is in the same square as the agent. The action *Shoot* can be used to fire an arrow in a straight line in the direction the agent is facing. The arrow continues until it either hits (and hence kills) the wumpus or hits a wall. The agent has only one arrow, so only the first *Shoot* action has any effect. Finally, the action *Climb* can be used to climb out of the cave, but only from square [1,1].

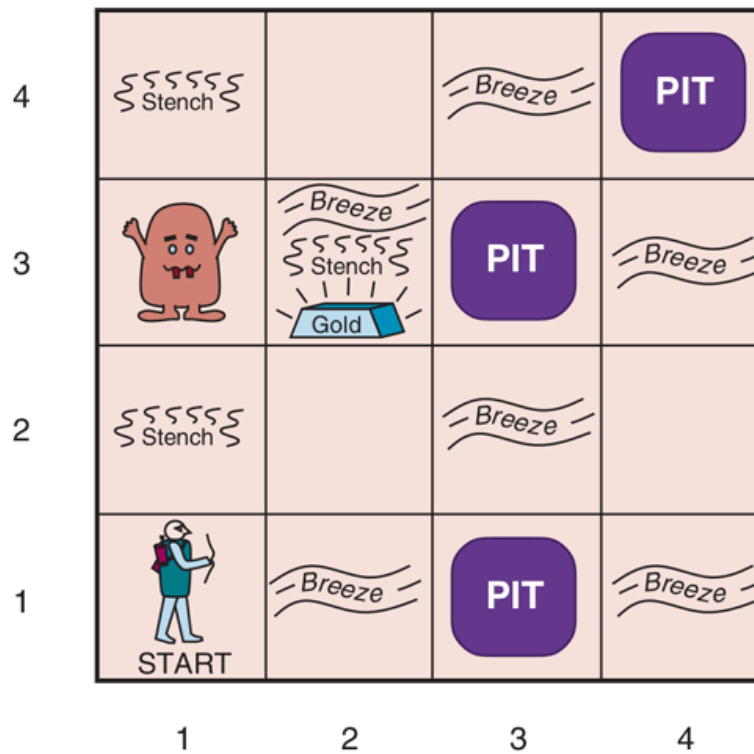
- **SENSORS:** The agent has five sensors, each of which gives a single bit of information:
 - In the squares directly (not diagonally) adjacent to the wumpus, the agent will perceive a *Stench*.¹

¹ Presumably the square containing the wumpus also has a stench, but any agent entering that square is eaten before being able to perceive anything.

- 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 anywhere in the cave.

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

Figure 7.2



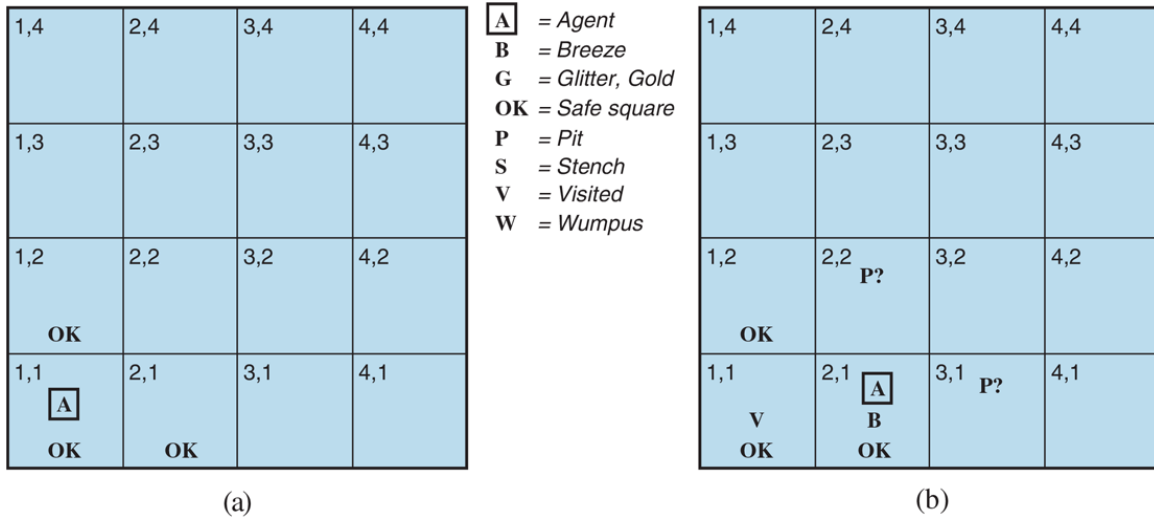
A typical wumpus world. The agent is in the bottom left corner, facing east (rightward).

We can characterize the wumpus environment along the various dimensions given in [Chapter 2](#). Clearly, it is deterministic, discrete, static, and single-agent. (The wumpus doesn't move, fortunately.) It is sequential, because rewards may come only after many actions are taken. It is partially observable, because some aspects of the state are not directly perceivable: the agent's location, the wumpus's state of health, and the availability of an arrow. As for the locations of the pits and the wumpus: we could treat them as unobserved parts of the state—in which case, the transition model for the environment is completely known, and finding the locations of pits completes the agent's knowledge of the state. Alternatively, we could say that the transition model itself is unknown because the agent doesn't know which *Forward* actions are fatal—in which case, discovering the locations of pits and wumpus completes the agent's knowledge of the transition model.

For an agent in the environment, the main challenge is its initial ignorance of the configuration of the environment; overcoming this ignorance seems to require logical reasoning. In most instances of the wumpus world, it is possible for the agent to retrieve the gold safely. Occasionally, the agent must choose between going home empty-handed and risking death to find the gold. About 21% of the environments are utterly unfair, because the gold is in a pit or surrounded by pits.

Let us watch a knowledge-based wumpus agent exploring the environment shown in [Figure 7.2](#). We use an informal knowledge representation language consisting of writing down symbols in a grid (as in [Figures 7.3](#) and [7.4](#)).

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 moving to `[2, 1]` and perceiving `[None, Breeze, None, None, None]`.

Figure 7.4



Two later stages in the progress of the agent. (a) After moving to `[1, 1]` and then `[1, 2]`, and perceiving `[Stench, None, None, None, None]`. (b) After moving to `[2, 2]` and then `[2, 3]`, and perceiving `[Stench, Breeze, Glitter, None, None]`.

The agent's initial knowledge base contains the rules of the environment, as described previously; in particular, it knows that it is in [1,1] and that [1,1] is a safe square; we denote that with an "A" and "OK," respectively, in square [1,1].

The first percept is *[None, None, None, None, None]*, from which the agent can conclude that its neighboring squares, [1,2] and [2,1], are free of dangers—they are OK. Figure 7.3(a) shows the agent's state of knowledge at this point.

A cautious agent will move only into a square that it knows to be OK. Let us suppose the agent decides to move forward to [2,1]. The agent perceives a breeze (denoted by "B") 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 that has not yet been visited. So the prudent agent will turn around, go back to [1,1], and then proceed to [1,2].

The agent perceives a stench in [1,2], 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 inference. Moreover, the lack of a breeze in [1,2] implies that there is no pit in [2,2]. Yet the agent has 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 relies on the lack of a percept to make one crucial step.

The agent has now proved to itself that there is neither a pit nor a wumpus in [2,2], so it is OK to move there. We do not show the agent's state of knowledge at [2,2]; we just assume that the agent turns and moves to [2,3], giving us Figure 7.4(b). In [2,3], the agent detects a glitter, so it should grab the gold and then return home.

Note that in each case for which the agent draws a conclusion from the available information, that conclusion is *guaranteed* to be correct if the available information is correct. This is a fundamental property of logical reasoning. In the rest of this chapter, we

describe how to build logical agents that can represent information and draw conclusions such as those described in the preceding paragraphs.