**Logical agents**

The central component of a knowledge-based agent is its **knowledge base**, or KB. A knowledge base is a set of **sentences**. (Here “sentence” is used as a technical term. It is related but not identical to the sentences of English and other natural languages.) Each sentence is expressed in a language called a **knowledge representation language** and represents some assertion about the world. Sometimes we dignify a sentence with the name **axiom**, when the sentence is taken as given without being derived from other sentences. There must be a way to add new sentences to the knowledge base and a way to query what is known. The standard names for these operations are TELL and ASK, respectively. Both operations may involve **inference**—that is, deriving new sentences from old. Inference must obey the requirement that when one ASKs a question of the knowledge base, the answer should follow from what has been told to the knowledge base previously. Later we will be more precise about the crucial word “follow.” For now, take it to mean that the inference process should not make things up as it goes along. Figure 7.1 shows the outline of a knowledge-based agent program. Like all our agents, it takes a percept as input and returns an action. The agent maintains a knowledge base, KB, which may initially contain some **background knowledge**.

Each time the agent program is called, it does three things. First, it TELLs the knowledge

base what it perceives. Second, it ASKs the knowledge base what action it should perform. In the process of answering this query, extensive reasoning may be done about the current state of the world, about the outcomes of possible action sequences, and so on. Third, the agent program TELLs the knowledge base which action was chosen, and the agent executes the action.

The details of the representation language are hidden inside three functions that implement

the interface between the sensors and actuators on one side and the core representation and reasoning system on the other. MAKE-PERCEPT-SENTENCE constructs a sentence asserting that the agent perceived the given percept at the given time. MAKE-ACTION-QUERY

constructs a sentence that asks what action should be done at the current time. Finally, MAKE-ACTION-SENTENCE constructs a sentence asserting that the chosen action was executed. The details of the inference mechanisms are hidden inside TELL and ASK. Later

sections will reveal these details.

The agent in Figure 7.1 appears quite similar to the agents with internal state. Because of the definitions of TELL and ASK, however, the knowledge-based agent is not an arbitrary program for calculating actions. It is amenable to a description at the **knowledge level**, where we need to specify only what the agent knows and what its goals are, in order to fix its behavior. For example, an automated taxi might have the goal of taking a passenger from San Francisco to Marin County and might know that the Golden Gate Bridge is the only link between the two locations. Then we can expect it to cross the Golden Gate Bridge *because it knows that that will achieve its goal*. Notice that this analysis is independent of how the taxi works at the **implementation level**. It doesn’t matter whether its geographical knowledge is implemented as linked lists or pixel maps, or whether it reasons by manipulating strings of symbols stored in registers or by propagating noisy signals in a network of neurons. A knowledge-based agent can be built simply by TELLing it what it needs to know. Starting with an empty knowledge base, the agent designer can TELL sentences one by one until the agent knows how to operate in its environment. This is called the **declarative** approach to system building. In contrast, the **procedural** approach encodes desired behaviors directly as program code. In the 1970s and 1980s, advocates of the two approaches engaged in heated debates. We now understand that a successful agent often combines both declarative and procedural elements in its design, and that declarative knowledge can often be compiled into more efficient procedural code.

We can also provide a knowledge-based agent with mechanisms that allow it to learn for itself. These mechanisms, which are discussed in Chapter 18, create general knowledge about the environment from a series of percepts. A learning agent can be fully autonomous.



**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 mitigating 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.

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. The agent always starts in the square labeled [1,1], facing to the right. 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 square containing the wumpus and in the directly (not diagonally) adjacent 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 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].

We can characterize the wumpus environment along the various dimensions given in Chapter

2. Clearly, it is 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 that happen to be immutable—in which case, the transition model for the environment is completely known; or 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).

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.



