

Chapter 7

Logical Agents

In which we design agents that can form representations of a complex world, use a process of inference to derive new representations about the world, and use these new representations to deduce what to do.

Humans, it seems, know things; and what they know helps them do things. In AI, **knowledge-based agents** use a process of **reasoning** over an internal **representation** of knowledge to decide what actions to take.

Knowledge-based agents

Reasoning

Representation

The problem-solving agents of **Chapters 3** and **4** know things, but only in a very limited, inflexible sense. They know what actions are available and what the result of performing a specific action from a specific state will be, but they don't know general facts. A route-finding agent doesn't know that it is impossible for a road to be a negative number of kilometers long. An 8-puzzle agent doesn't know that two tiles cannot occupy the same space. The knowledge they have is very useful for finding a path from the start to a goal, but not for anything else.

The atomic representations used by problem-solving agents are also very limiting. In a partially observable environment, for example, a problem-solving agent's only choice for representing what it knows about the current state is to list all possible concrete states. I could give a human the goal of driving to a U.S. town with population less than 10,000, but to say that to a problem-solving agent, I could formally describe the goal only as an explicit set of the 16,000 or so towns that satisfy the description.

Chapter 6 introduced our first factored representation, whereby states are represented as assignments of values to variables; this is a step in the right direction, enabling some parts of the agent to work in a domain-independent way and allowing for more efficient algorithms. In this chapter, we take this step to its logical conclusion, so to speak—we develop **logic** as a general class of representations to support knowledge-based agents. These agents can combine and recombine information to suit myriad purposes. This can be far removed from the needs of the moment—as when a mathematician proves a theorem or an astronomer calculates the Earth's life expectancy. Knowledge-based agents can accept new tasks in the form of explicitly described goals; they can achieve competence quickly by being told or learning new knowledge about the environment; and they can adapt to changes in the environment by updating the relevant knowledge.

We begin in **Section 7.1** with the overall agent design. **Section 7.2** introduces a simple new environment, the wumpus world, and illustrates the operation of a knowledge-based agent without going into any technical detail. Then we explain the general principles of **logic** in **Section 7.3** and the specifics of **propositional logic** in **Section 7.4**. Propositional logic is a factored representation; while less expressive than **first-order logic** (**Chapter 8**), which is the canonical structured representation, propositional logic illustrates all the basic concepts of logic. It also comes with well-developed inference technologies, which we describe in **sections 7.5** and **7.6**. Finally, **Section 7.7** combines the concept of knowledge-based agents with the technology of propositional logic to build some simple agents for the wumpus world.

7.1 Knowledge-Based 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. When the sentence is taken as being given without being derived from other sentences, we call it an **axiom**.

Knowledge base

Sentence

Knowledge representation language

Axiom

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 (or **Telled**) to the knowledge base previously. Later in this chapter, 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.

Inference


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**.

Figure 7.1

```
function KB-AGENT(percept) returns an action  
  persistent: KB, a knowledge base  
             t, a counter, initially 0, indicating time  
  
  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))  
  action ← ASK(KB, MAKE-ACTION-QUERY(t))  
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))  
  t ← t + 1  
  return action
```

A generic knowledge-based agent. Given a percept, the agent adds the percept to its knowledge base, asks the knowledge base for the best action, and tells the knowledge base that it has in fact taken that action.

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 returns the action so that it can be executed.

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 described in [Chapter 2](#). 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 specify only what the agent knows and what its goals are, in order to determine its behavior.

Knowledge level

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.

Implementation level

A knowledge-based agent can be built simply by `TELL`ing 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.

Declarative

Procedural

We can also provide a knowledge-based agent with mechanisms that allow it to learn for itself. These mechanisms, which are discussed in [Chapter 19](#), create general knowledge about the environment from a series of percepts. A learning agent can be fully autonomous.