LOGICAL AGENTS

Knowledge-based agents

- Knowledge based agents use a process of reasoning over an internal representation of knowledge to decide what actions to take.

- Knowledge-based agents can accept new tasks in the form of explicitly described goals;
  - Can achieve competence quickly by being told or learning new knowledge about the environment;
  - Can adapt to changes in the environment by updating the relevant knowledge.

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**.

- Each **sentence** is expressed in a language called a **knowledge representation language** and represents some assertion about the world.

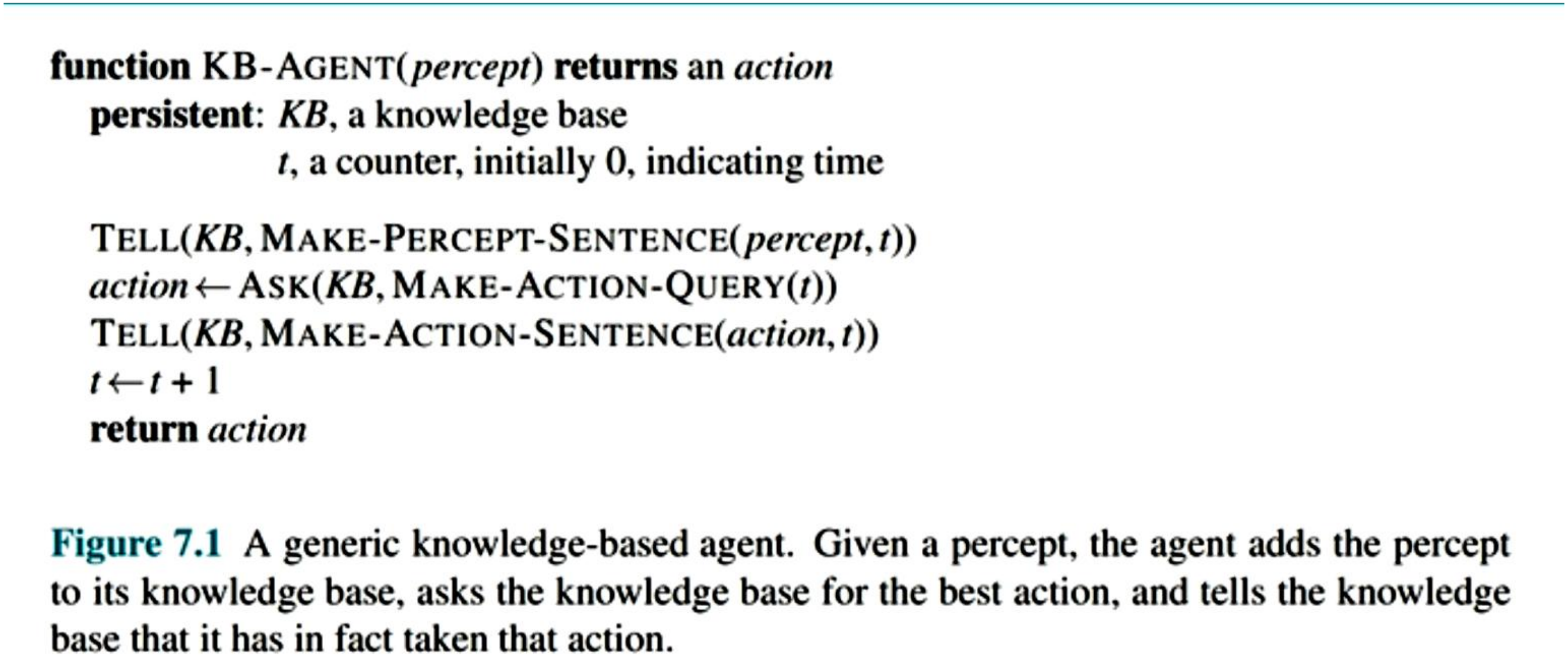
- When the sentence is taken as being given without being derived from other sentences, we call it an **axiom**.

- TELL and ASK, are to add new sentences to the knowledge base and a way to query what is known, respectively.

- Both operations 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.

- KB may initially contain some background knowledge.



```
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
```

Figure 7.1 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.

- 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 behaviours directly as program code.