Artificial Intelligence

Intelligent Agent

Sadullah Karimi

MSc in Computer Science and Engineering

Assistant Professor at Avicenna University

Table of Contents

AGENTS AND ENVIRONMENTS

An agent is anything that can be viewed as perceiving its environment through sensors.

And acting upon that environment through actuators.

A robotic agent might have **cameras** and **infrared** range finders for sensors and various **motors** for actuators.

A software agent receives:

- Keystrokes,
- File contents,

and network packets as sensory inputs and acts on the environment by displaying on the

- Screen,
- Writing files,
- And sending network packets.

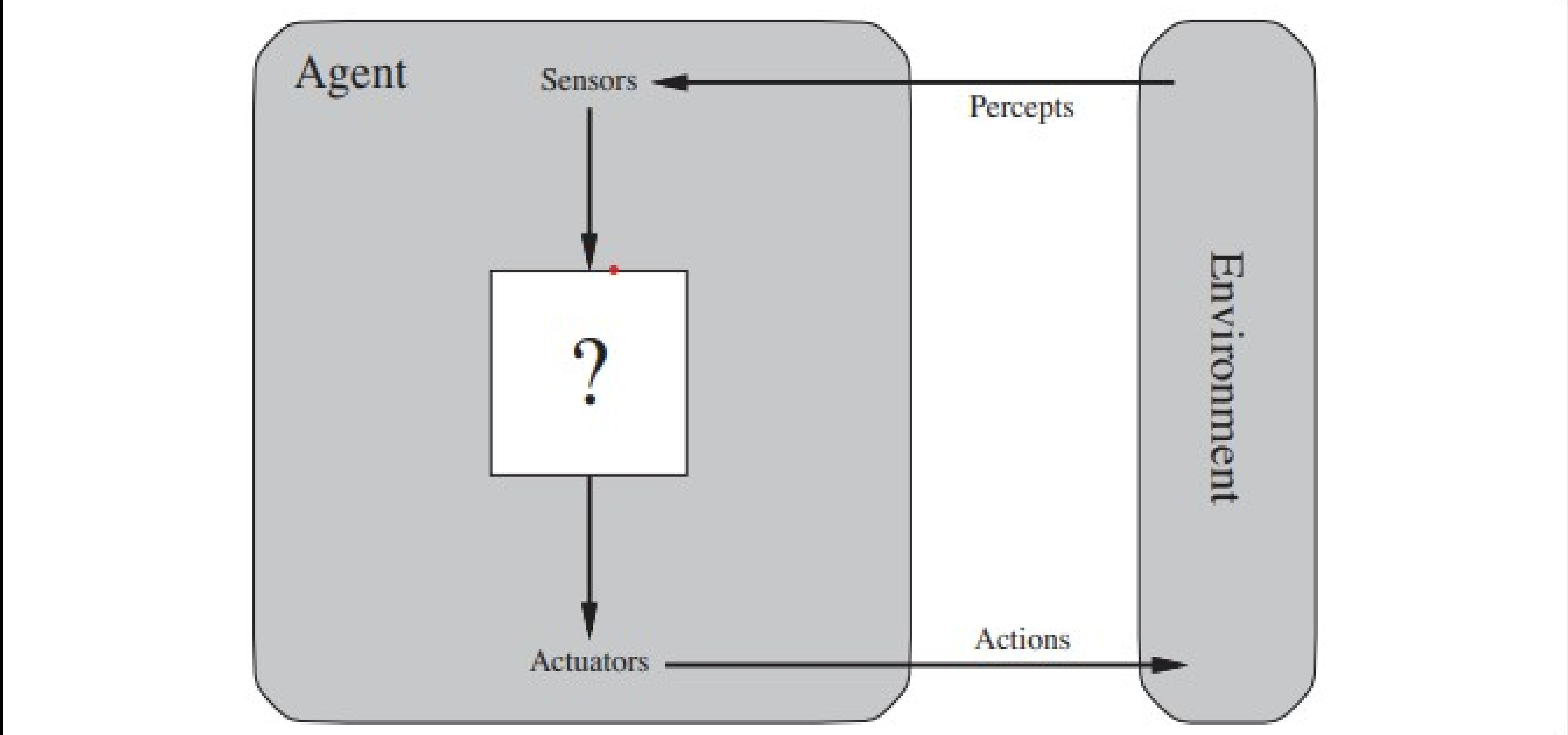


Figure 2.1 Agents interact with environments through sensors and actuators.

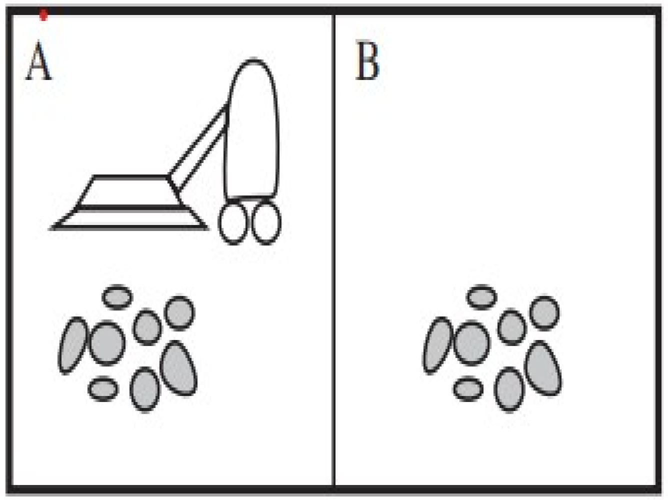
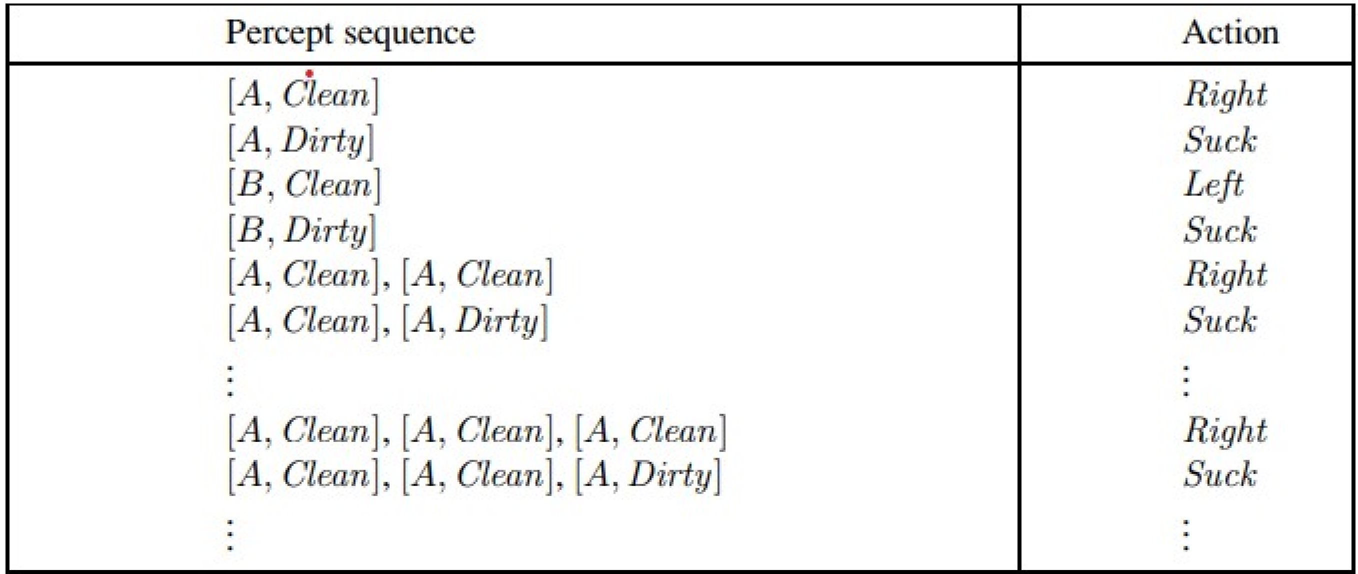


Figure 2.2 A vacuum-cleaner world with just two locations.



Percept sequence	Action
[*A*, *Clean*]	*Right*
[*A*, *Dirty*]	*Suck*
[*B*, *Clean*]	*Left*
[*B*, *Dirty*]	*Suck*
[*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Sequence of States and Desirability:

- The agent's actions lead the environment through a sequence of states.
- A sequence is considered desirable if it achieves a predefined goal or outcome that is beneficial to the environment or the task at hand.

Performance Measure

- A performance measure evaluates how well the agent performs over time.
- It quantifies the desirability of the sequence of states produced by the agent's actions.
- In your scenario, the initial performance measure was based on the amount of dirt cleaned up in an eight-hour shift. However, this led to sub optimal behavior (dumping dirt and re cleaning).
- A more suitable performance measure should incentivize behavior that aligns with the ultimate goal of having a clean floor, rather than just maximizing a simplistic metric like dirt cleaned.

Designing a Suitable Performance Measure:

- Cleanliness Metric: Award points for each clean square at each time step. This directly rewards the agent for achieving a clean floor.
- Penalties: Introduce penalties for actions that are undesirable, such as excessive electricity consumption or noise generation. These penalties discourage behaviors that are not aligned with the overall goal (cleaning efficiently and quietly).

Rational Agent Behavior:

- A rational agent aims to maximize its performance measure.
- Initially, based on the dirt quantity measure, the agent might exhibit sub-optimal behavior (dumping and re-cleaning).
- With a cleanliness-based performance measure, the agent would prioritize cleaning effectively and maintaining cleanliness to maximize its score.
- In summary, the key to designing an effective performance measure is to align it closely with the ultimate goal of the task. In a cleaning scenario, this means focusing on cleanliness metrics (clean squares) while considering and penalizing factors that detract from overall effectiveness (electricity consumption, noise). This approach encourages the agent to act in ways that are genuinely beneficial rather than exploiting loopholes in the measurement system.

Overall Goal:

- The goal is to design a performance measure that encourages behavior that directly contributes to achieving the desired outcome—in this case, a clean floor—while discouraging actions that detract from this goal.

Rationality

- ✓ What is rational at any given time depends on four things:
  - The performance measure that defines the criterion of success.
  - The agent’s prior knowledge of the environment.
  - The actions that the agent can perform.
  - The agent’s percept sequence to date.

or each possible percept sequence,

- ✓ a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the precept sequence and whatever built-in knowledge the agent has.

Omniscience, learning, and autonomy

- ✓ An omniscient agent knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality.
- ✓ Consider the following example:

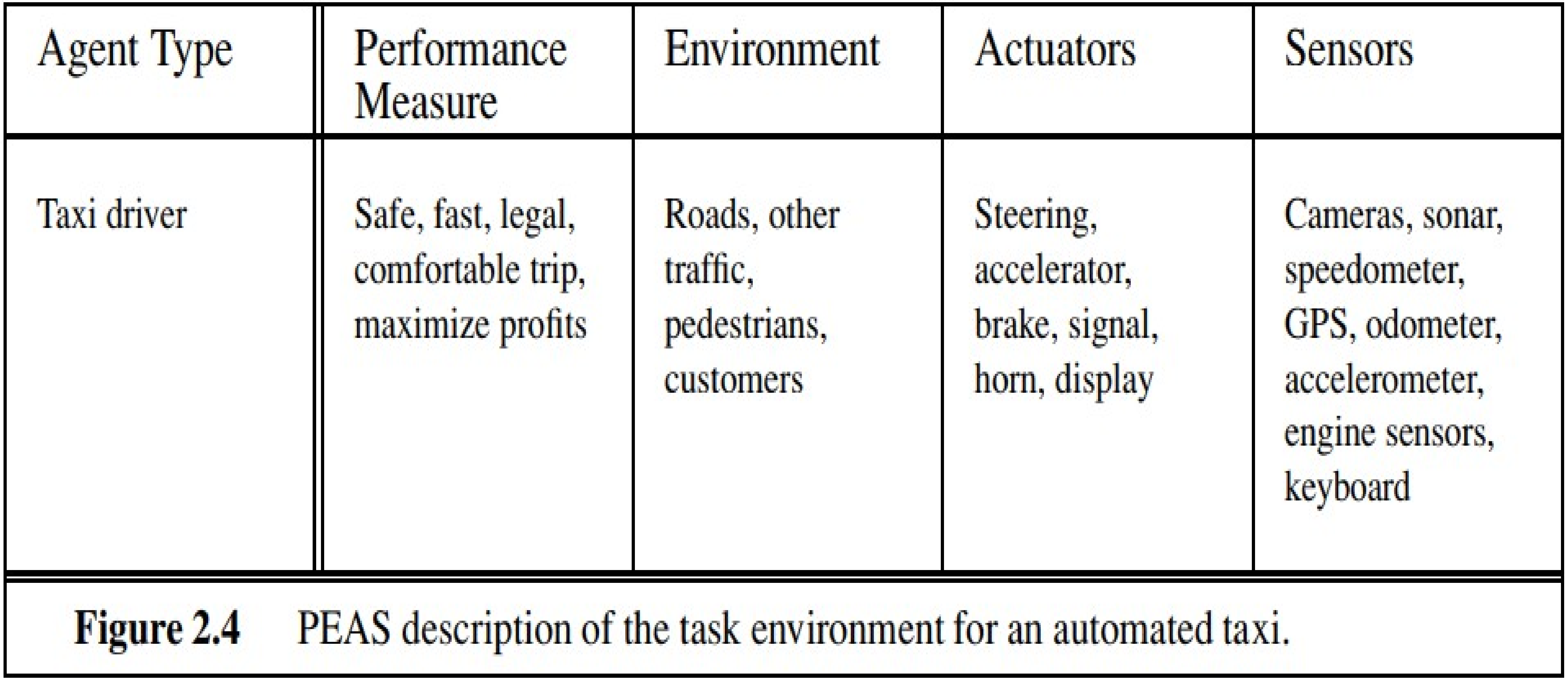
- I am walking along the Champs Elys´ees one day and I see an old friend across the street. There is no traffic nearby and I'm not otherwise engaged, so, being rational, I start to cross the street. Meanwhile, at 33,000 feet, a cargo door falls off a passing airliner,2 and before I make it to the other side of the street I am flattened. Was I irrational to cross the street? It is unlikely that my obituary would read "Idiot attempts to cross street." This example shows that rationality is not the same as perfection. Rationality maximizes expected performance, while perfection maximizes actual performance. Retreating from a requirement of perfection is not just a question of being fair to agents. The point is that if we expect an agent to do what turns out to be the best action after the fact, it will be impossible to design an agent to fulfill this specification—unless we improve the performance of crystal balls or time machines.

THE NATURE OF ENVIRONMENTS

✓ **TASK ENVIRONMENT:**

- Now that we have a definition of rationality, we are almost ready to think about building rational agents.
- First, however, we must think about task environments, which are essentially the "problems" to which rational agents are the "solutions".
- We begin by showing how to specify a task environment, illustrating the process with a number of examples.
- We then show that task environments come in a variety of flavors. The flavor of the task environment directly affects the appropriate design for the agent program.

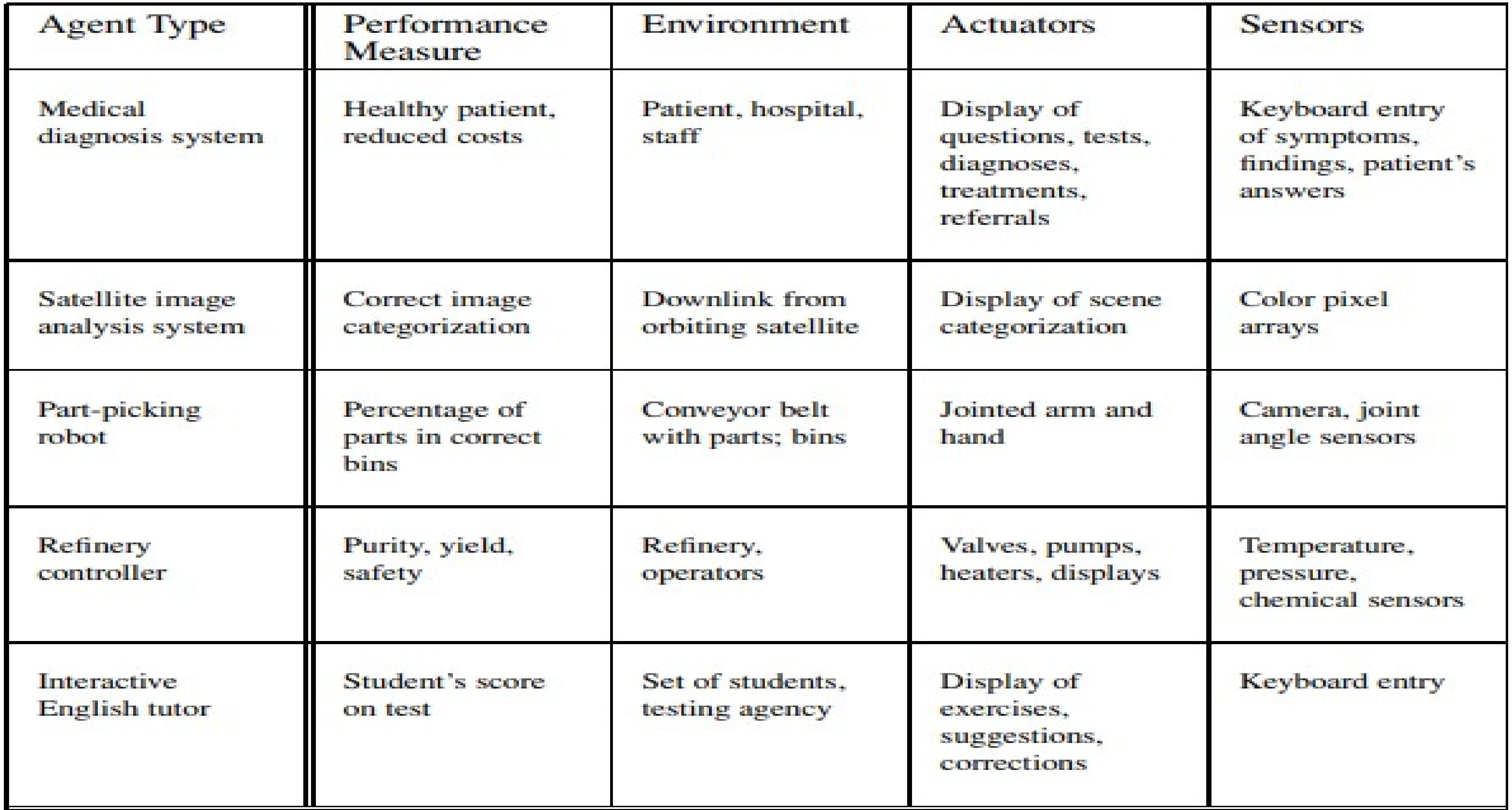
Specifying the task environment



Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

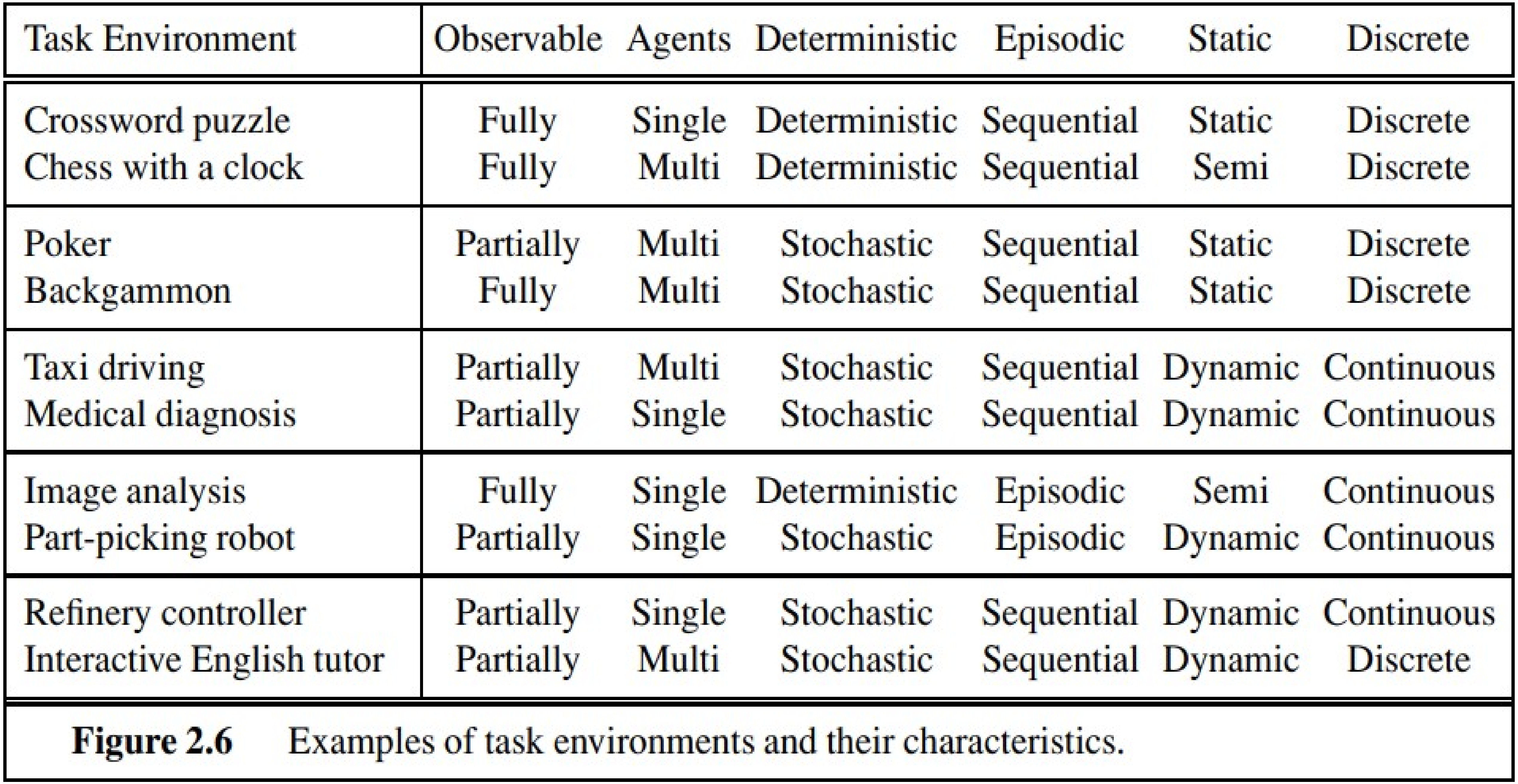
Figure 2.4 PEAS description of the task environment for an automated taxi.

Properties of task environments



Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Figure 2.5 Examples of agent types and their PEAS descriptions.



Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Figure 2.6 Examples of task environments and their characteristics.

THE STRUCTURE OF AGENTS

Agent = architecture + program

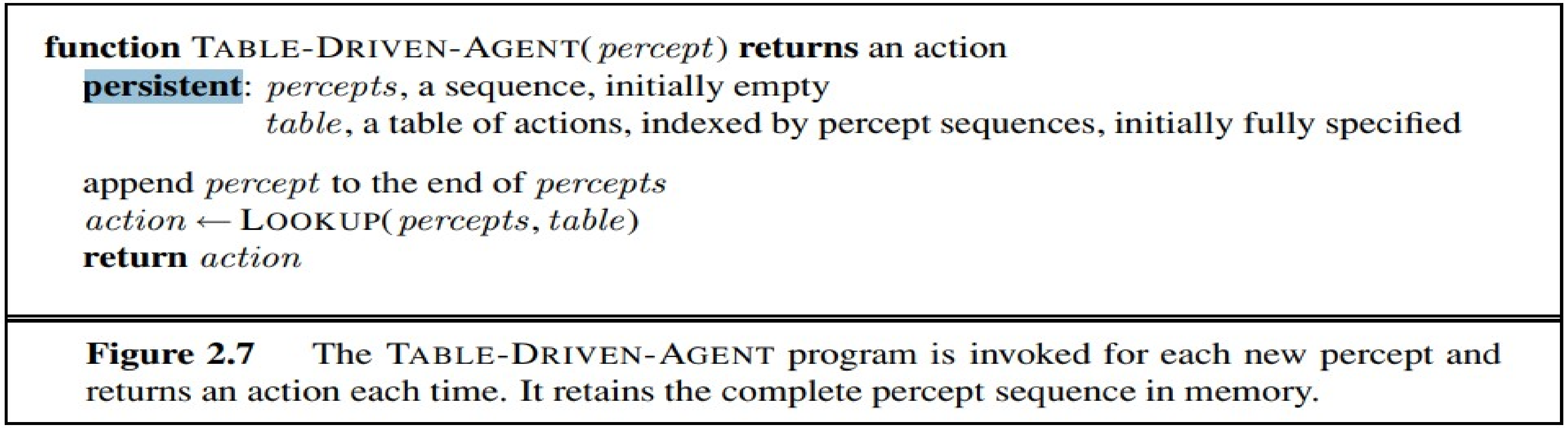
Obviously, the program we choose has to be one that is appropriate for the architecture. If the program is going to recommend actions like Walk, the architecture had better have legs.

The architecture might be just an ordinary PC, or it might be a robotic car with several onboard computers, cameras, and other sensors.

In general, the architecture makes the percepts from the sensors available to the program, runs the program, and feeds the program's action choices to the actuators as they are generated.

Most of this book is about designing agent programs, although Chapters 24 and 25 deal directly with the sensors and actuators.

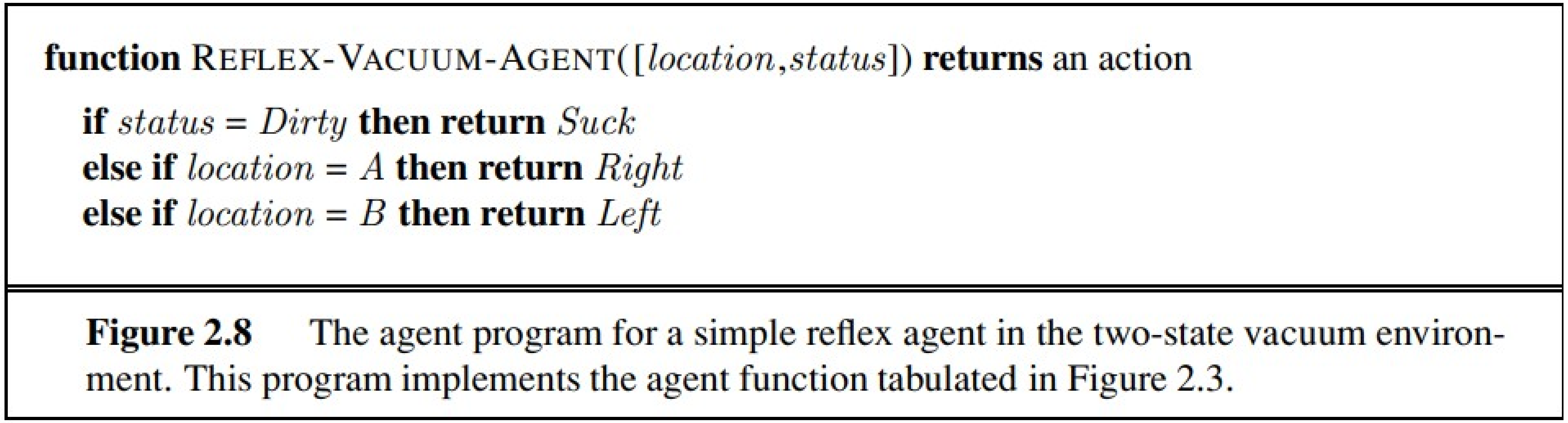
Agent programs



```
function TABLE-DRIVEN-AGENT(percept) returns an action
    persistent: percepts, a sequence, initially empty
                table, a table of actions, indexed by percept sequences, initially fully specified

    append percept to the end of percepts
    action ← LOOKUP(percepts, table)
    return action
```

Figure 2.7 The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.



```
function REFLEX-VACUUM-AGENT([location,status]) returns an action

    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

Figure 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

Simple reflex agents

- ✓ CONDITION–ACTION RULE

- Simple reflex behaviors occur even in more complex environments.
- Imagine yourself as the driver of the automated taxi.
- If the car in front brakes and its brake lights come on, then you should notice this and initiate braking.
- In other words, some processing is done on the visual input to establish the condition we call “The car in front is braking.”
- Then, this triggers some established connection in the agent program to the action “initiate braking.” We call such a connection a condition–action rule, written as:
    - if car-in-front-is-braking then initiate-braking.
    - Also called situation–action rules, productions, or if–then rules.
    - The agent function only succeeds when the environment is fully observable.

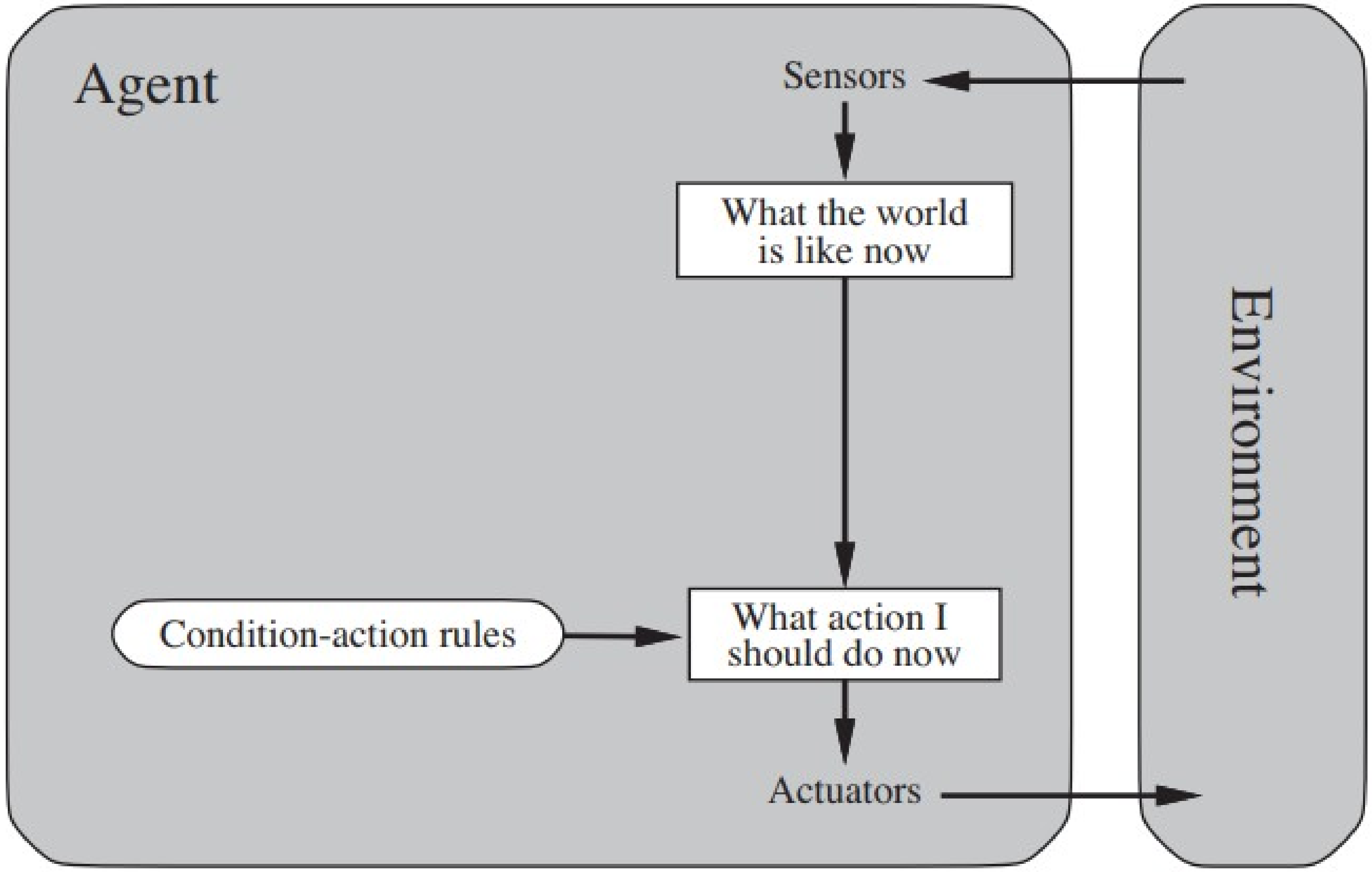
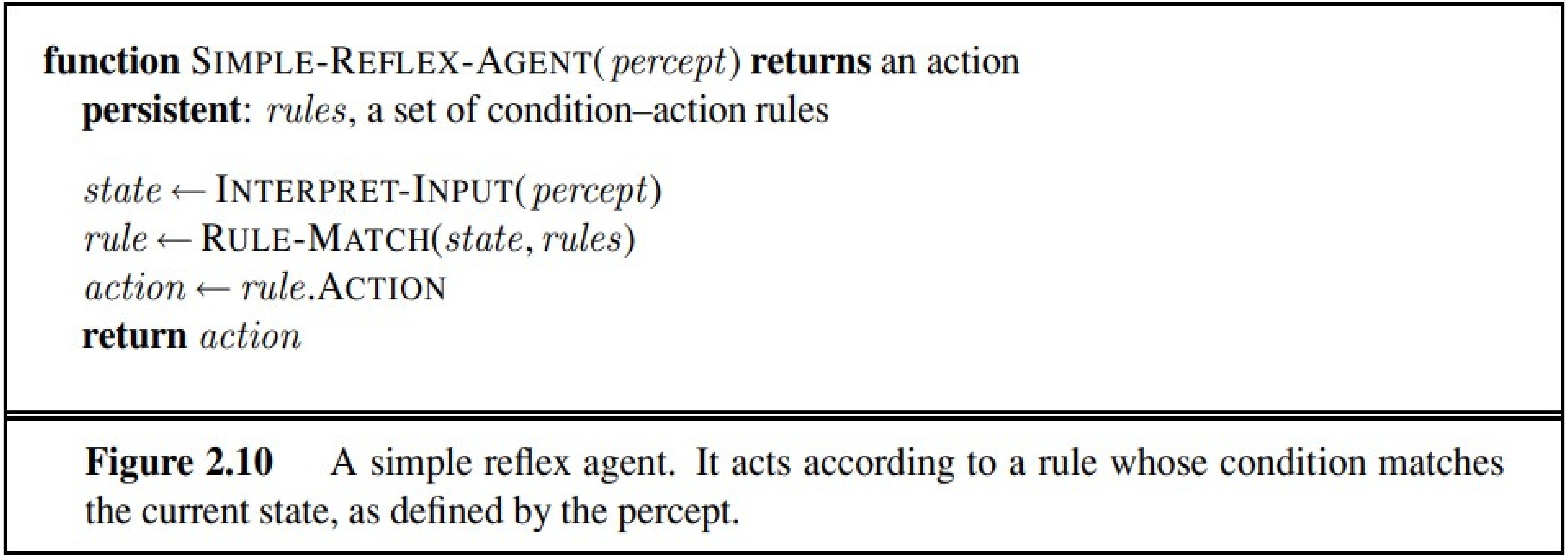


Figure 2.9 Schematic diagram of a simple reflex agent.



```
function SIMPLE-REFLEX-AGENT(percept) returns an action
    persistent: rules, a set of condition–action rules

    state ← INTERPRET-INPUT(percept)
    rule ← RULE-MATCH(state, rules)
    action ← rule.ACTION
    return action
```

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

- ✓ The RULE-MATCH function returns the first rule in the set of rules that matches the given state description.
- ✓ Infinite loops are often unavoidable for simple reflex agents operating in partially observable environments.
- ✓ RANDOMIZATION
  - Escape from infinite loops is possible if the agent can randomize its actions.

Problems with Simple reflex agents are:

- Very limited intelligence.
- No knowledge of non-perceptual parts of the state.
- Usually to big to generate and store.
- If there occurs any change in the environment, the the collection of rules need to be updated.

Model-based reflex agents

- ➔ **INTERNAL STATE**
  - maintain some sort of internal state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.
  - Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program.
  - for example, that an overtaking car generally will be closer behind than it was a moment ago.

➔This knowledge about "how the world works"—whether implemented in simple Boolean circuits or in complete scientific theories—is called a model of the world. An agent that uses such a model is called a model-based agent.

➔gives the structure of the model-based reflex agent with internal state, showing how the current percept is combined with the old internal state to generate the updated description of the current state, based on the agent's model of how the world works.

➔For example, an automated taxi may not be able to see around the large truck that has stopped in front of it and can only guess about what may be causing the hold-up. Thus, uncertainty about the current state may be unavoidable, but the agent still has to make a decision.

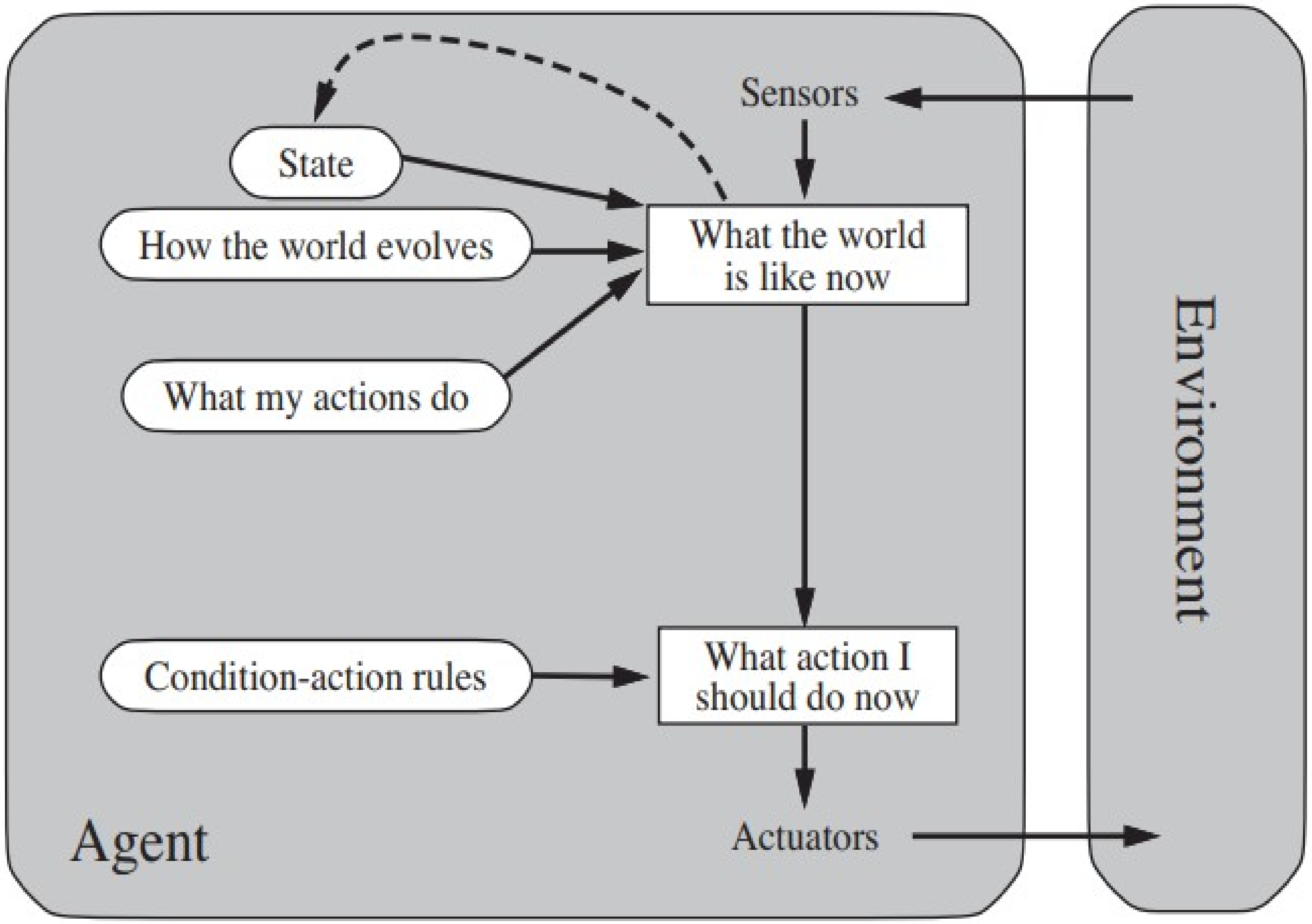
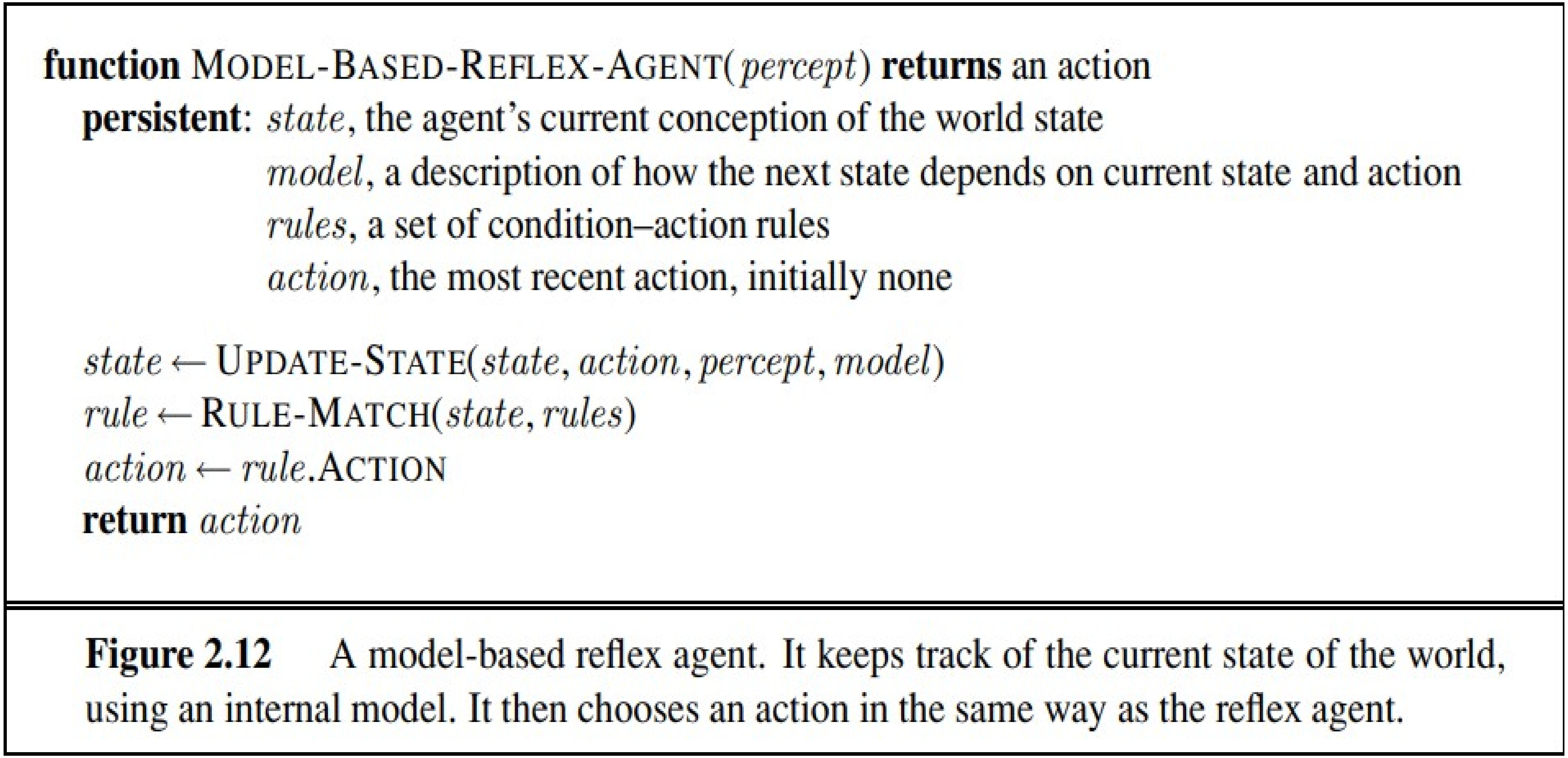


Figure 2.11 A model-based reflex agent.



```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

Goal-based agents

- The agent program can combine this with the model (the same information as was used in the model based reflex agent) to choose actions that achieve the goal.
- Based on:
  - Search
  - Planning
- It is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified

- Example, the taxi may be driving back home, and it may have a rule telling it to fill up with gas on the way home unless it has at least half a tank. Although “driving back home” may seem to an aspect of the world state, the fact of the taxi’s destination is actually an aspect of the agent’s internal state. If you find this puzzling, consider that the taxi could be in exactly the same place at the same time, but intending to reach a different destination.

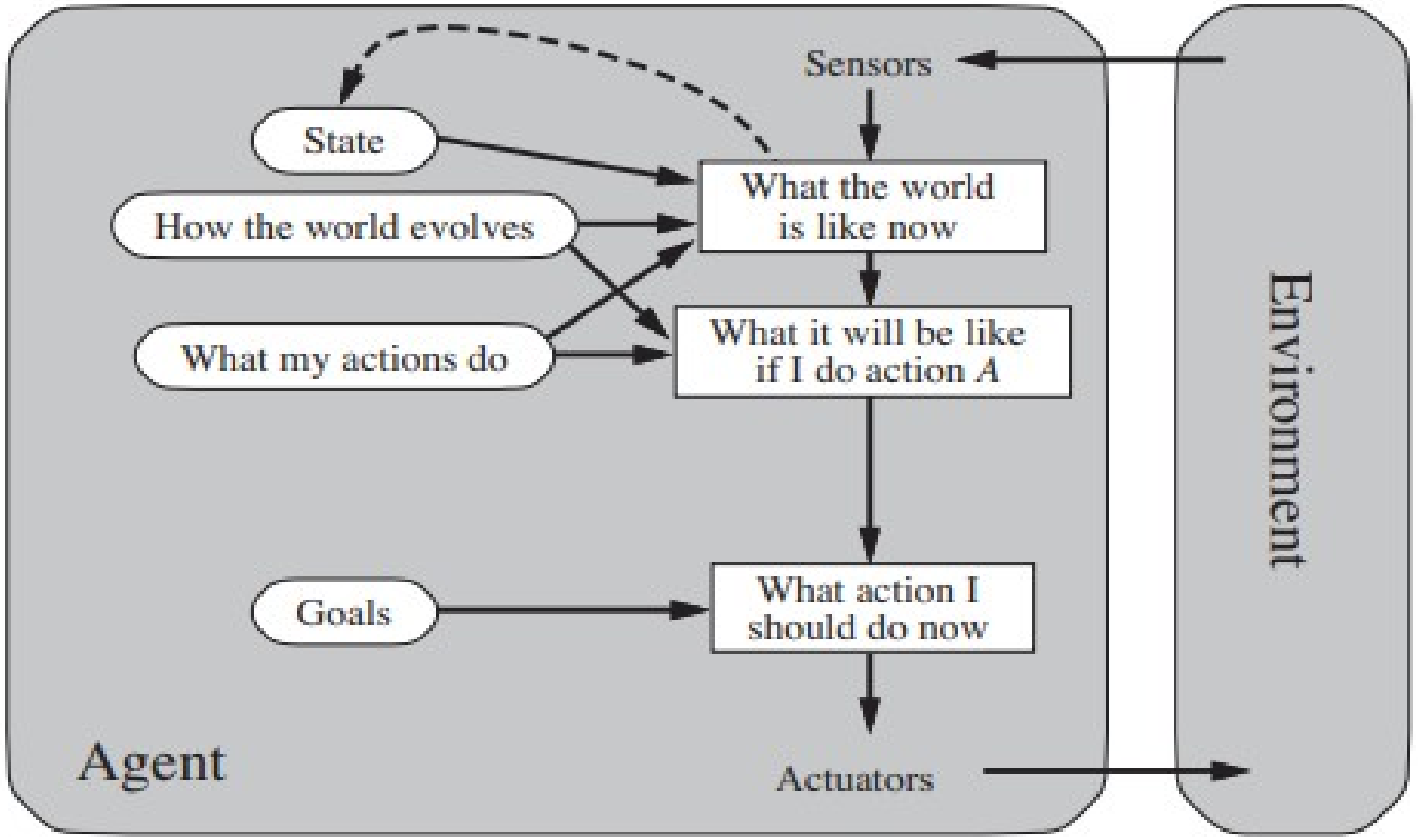

Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Utility-based agents

- **Definition**: Agents that prioritize outcomes based on utility functions or preferences.
- **Components:**
  - Utility Function: Defines preferences over possible outcomes.
  - Decision Making: Selects actions to maximize expected utility.
  - Rationality: Balances trade-offs for optimal decisions.
- **Applications:**
  - Healthcare: Treatment planning based on patient preferences and medical outcomes.
  - Finance: Investment decisions considering risk, return, and investor preferences.
  - Robotics: Path planning for robots based on safety, efficiency, and task completion.
- **Challenges:**
  - Complexity: Designing accurate utility functions reflecting human preferences.
  - Ethics: Ensuring decisions align with societal values and ethical norms.
  - Computational Limits: Real-time decision-making requires substantial computational resources.

- **Future Directions:**
  - Ethical AI: Developing frameworks for ethical utility-based decision-making.
  - Advanced AI: Enhancing capabilities with deep learning and reinforcement learning.
- **Conclusion:** Utility-based agents provide powerful decision-making tools across diverse domains, with ongoing research refining their effectiveness and ethical implications.

s

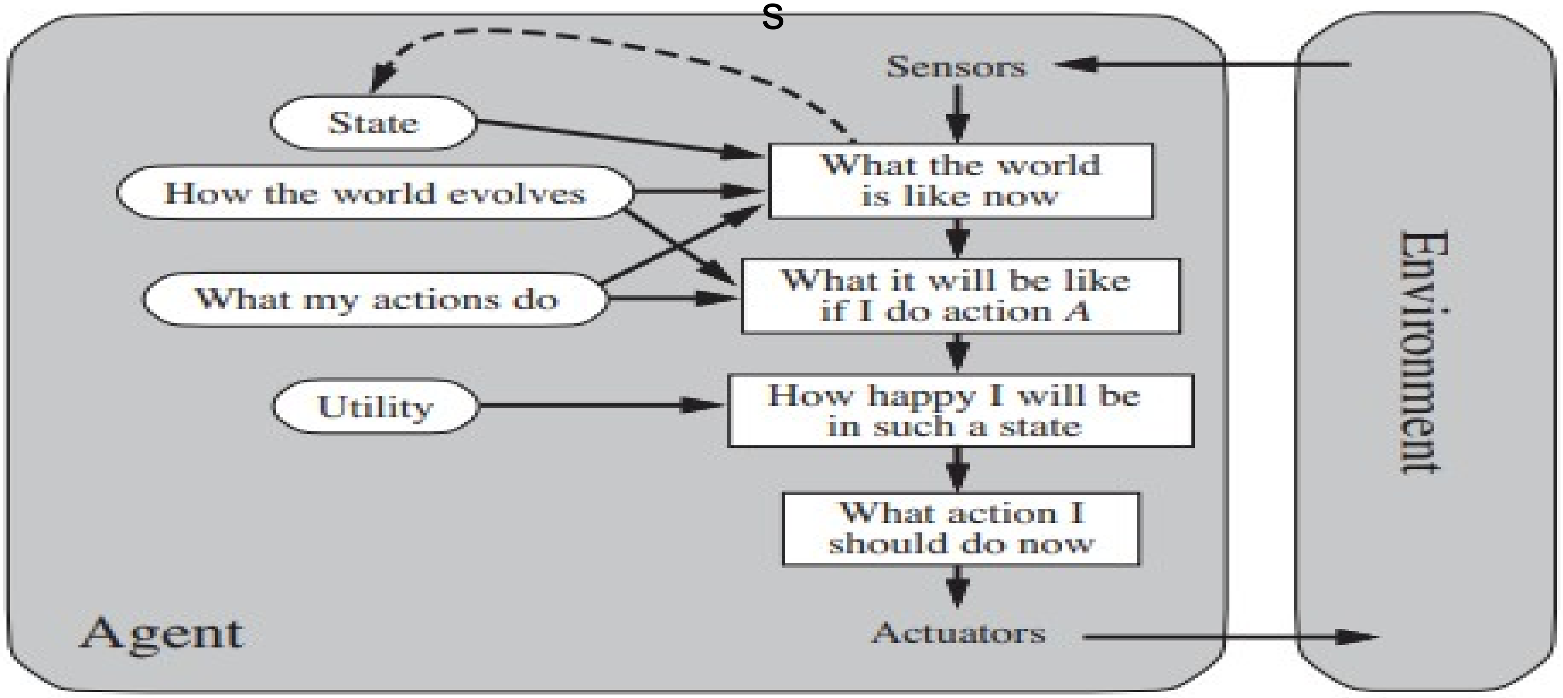

Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action A
What my actions do
Utility
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

Learning agents

- **Definition:** An agent that improves its performance based on experience and interactions with its environment.
- **Components:**
  - Performance Element: Determines actions taken by the agent.
  - Learning Element: Modifies the agent's behavior based on experience.
  - Critic: Provides feedback to evaluate the agent's actions.
  - Problem Generator: Suggests actions to explore new behaviors.
- **Types of Learning:**
  - Supervised Learning: Learns from labeled data.
  - Unsupervised Learning: Extracts patterns from unlabeled data.
  - Reinforcement Learning: Learns by trial and error through rewards and penalties.

Learning agent (Continue)

- **Applications:**
    - Autonomous Vehicles: Learn to navigate roads and respond to traffic.
    - Recommendation Systems: Personalize recommendations based on user preferences.
    - Game Playing: Improve strategies in games through experience.
- **Challenges:**
    - Data Quality: Requires large, high-quality datasets for effective learning.
    - Algorithm Complexity: Developing efficient algorithms for learning.
    - Ethical Considerations: Ensuring fairness and bias mitigation in decision-making.
- **Future Directions:**
    - Deep Learning: Enhancing agent capabilities with deep neural networks.
    - Explainable AI: Improving transparency and interpretability of learning processes.
- **Conclusion**: Learning agents are pivotal in advancing AI capabilities, with ongoing research addressing challenges and expanding applications.

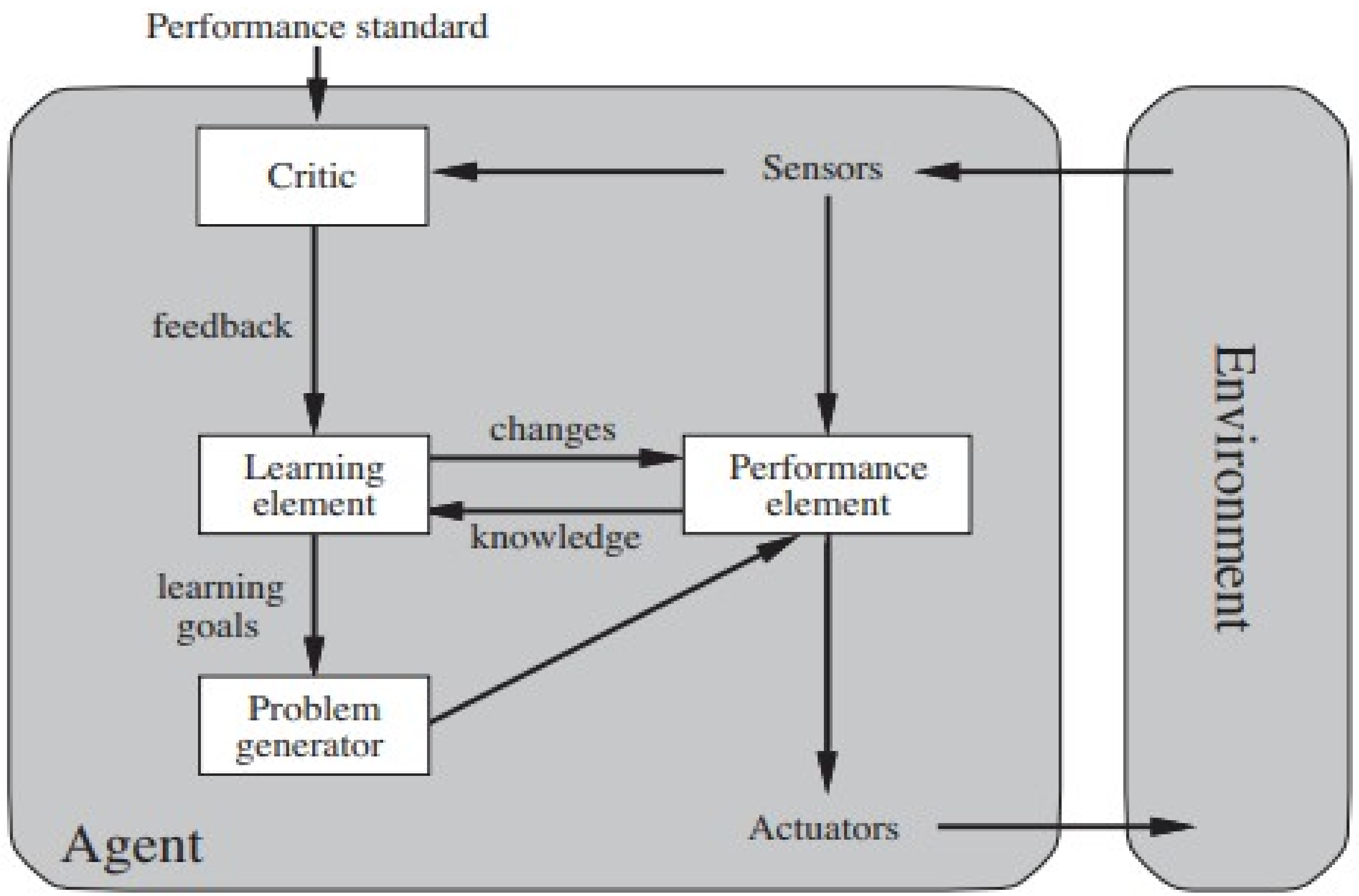


Figure 2.15 A general learning agent.

Components of Agent Programs

- ✓ **Atomic**
  - Definition: Simplest form where actions are indivisible.
  - Example: Basic reflex agents respond directly to stimuli without internal state.
- ✓ **Factored**
  - Definition: Actions and decisions are based on structured variables or features.
  - Example: Model-based reflex agents use internal state and current perceptions to decide actions.
- ✓ **Structured**
  - Definition: Complex agents with hierarchical decision-making and planning capabilities.
  - Example: Goal-based agents maintain goals and plans to achieve them over time.

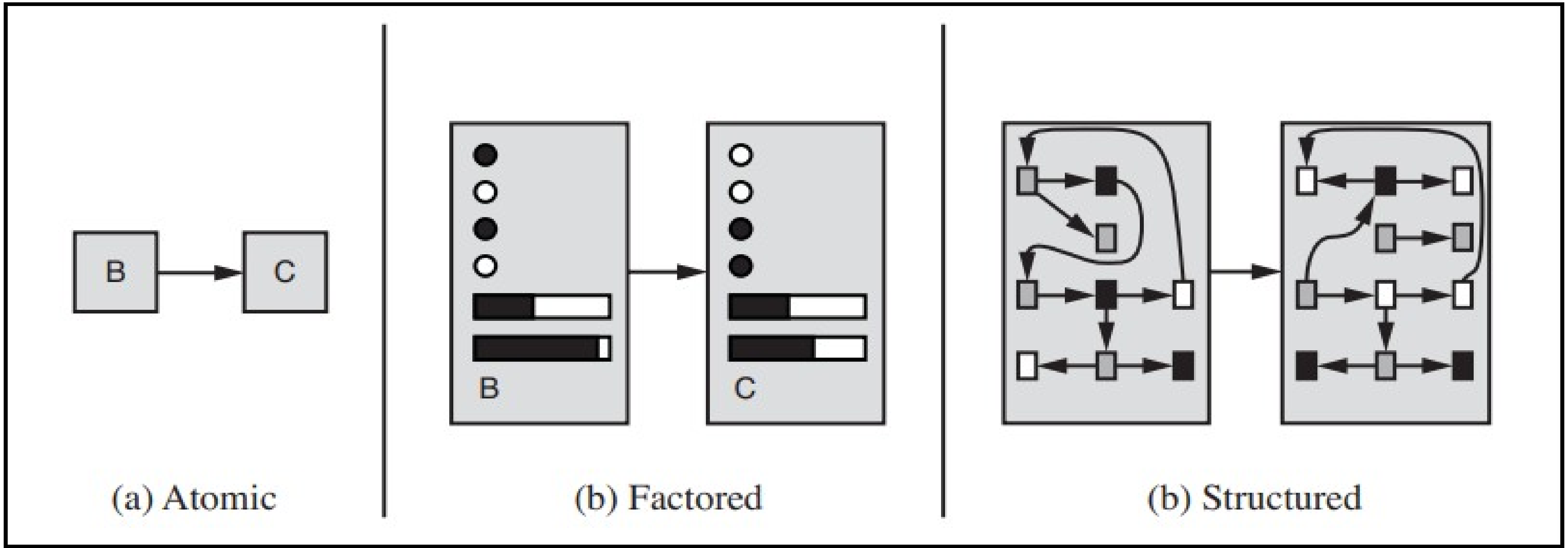


Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols. (c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

Thanks for your attention

Any questions?