Intelligent Agents

Agents and Environment

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

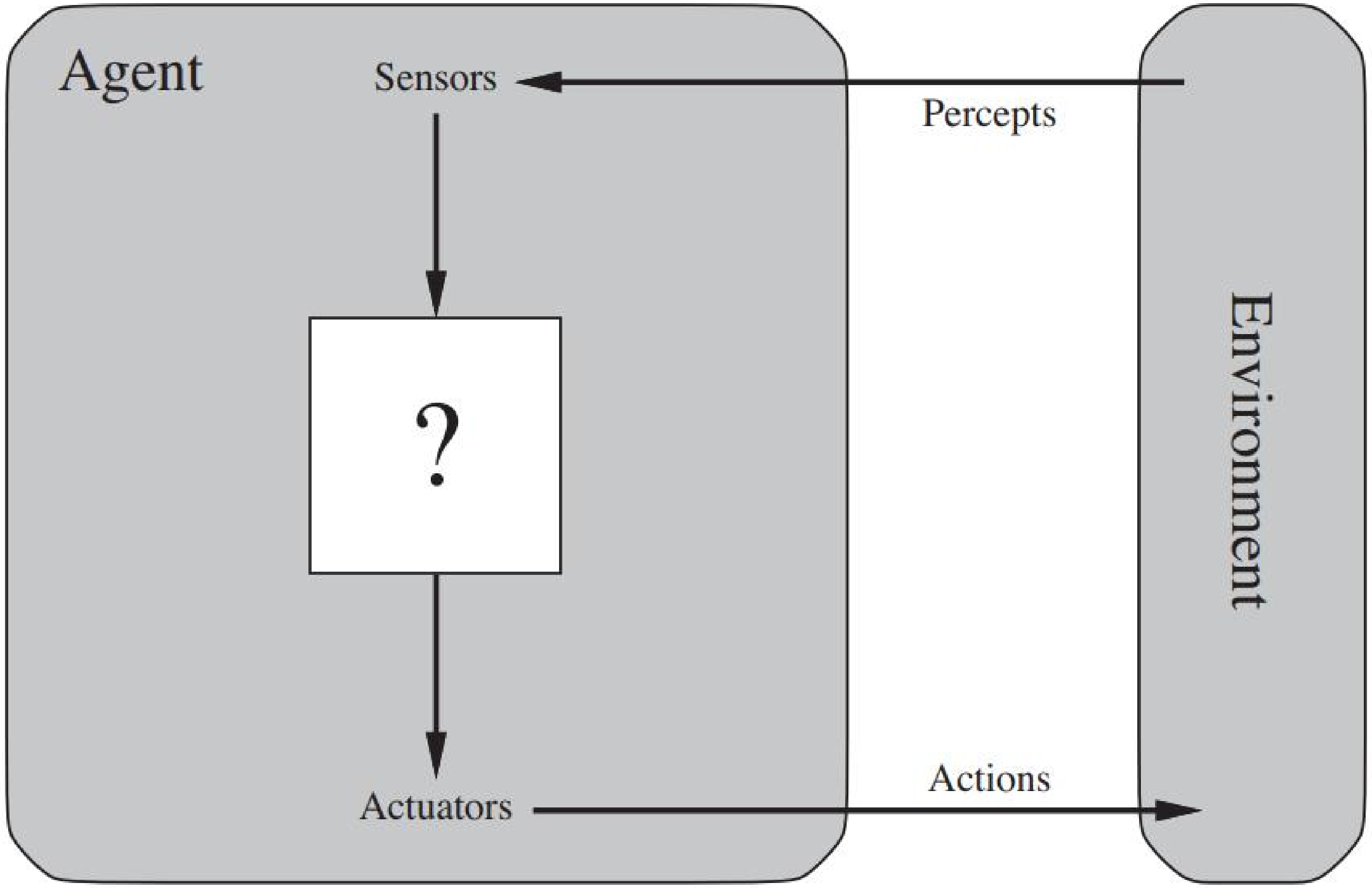
We use the term percept to refer to the agent's perceptual inputs at any given instant.

The agent's behavior is described by the agent function that maps any given percept sequence to an action.

The agent function for an artificial agent can be implemented by agent program.

The agent function is abstract mathematical description; the agent program is a concrete implementation, running within some physical system.

Agents and Environment



Agent Function

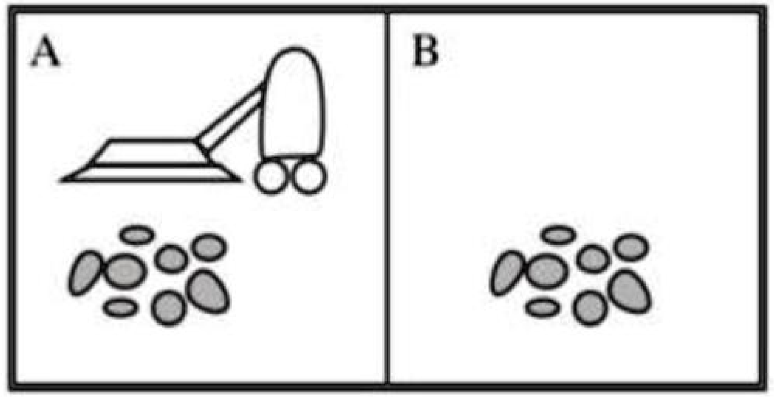


Figure 2.2 A vacuum-cleaner world with just two locations.

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.

Agents Program

Example: Vacuum-Cleaner World and Agent

Example 6.2.7 (Agent Program).

```
procedure 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
```

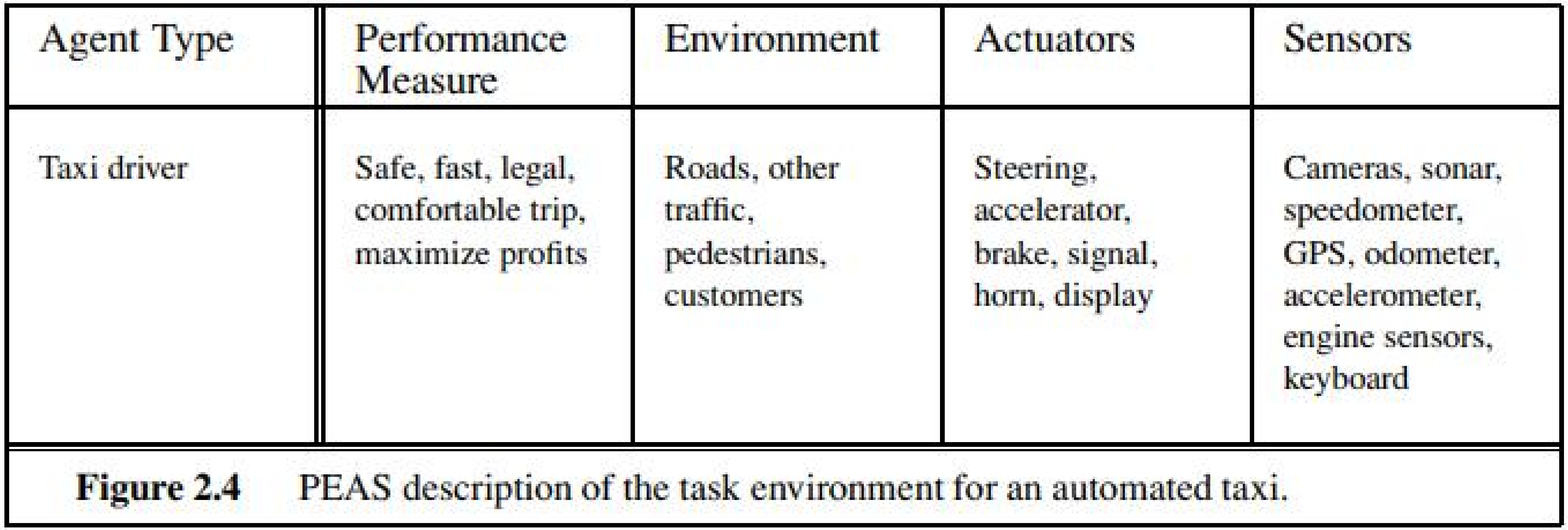
The Nature of Environments

In our discussion of the rationality of the simple vacuum-cleaner agent, we had to specify the performance measure, the environment and the agent's actuators and sensors.

We group all these under the heading of the task environment and also called as PEAS (Performance, Environment, Actuators, Sensors).

The vacuum world was a simple example; let us consider a more complex problem; an automated taxi driver.

Specifying the task environments



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.

The Nature of Environments

First, what is the performance measure to which we would like our automated driver to aspire? (Correct destination, minimizing fuel)

What is the driving environment that the taxi will face? (Roads, other traffic, pedestrians)

The actuators for an automated taxi include those available to a human driver: control over the engine through the accelerator and control over steering and braking.

The basic sensors for the taxi will include one or more controllable video cameras so that it can see the road; it might augment these with infrared or sonar sensors to detect distances to other cars and obstacles.

The Structure of Agents

Four basic agents:

a. Simple Reflex Agents

b. Model-based reflex agents

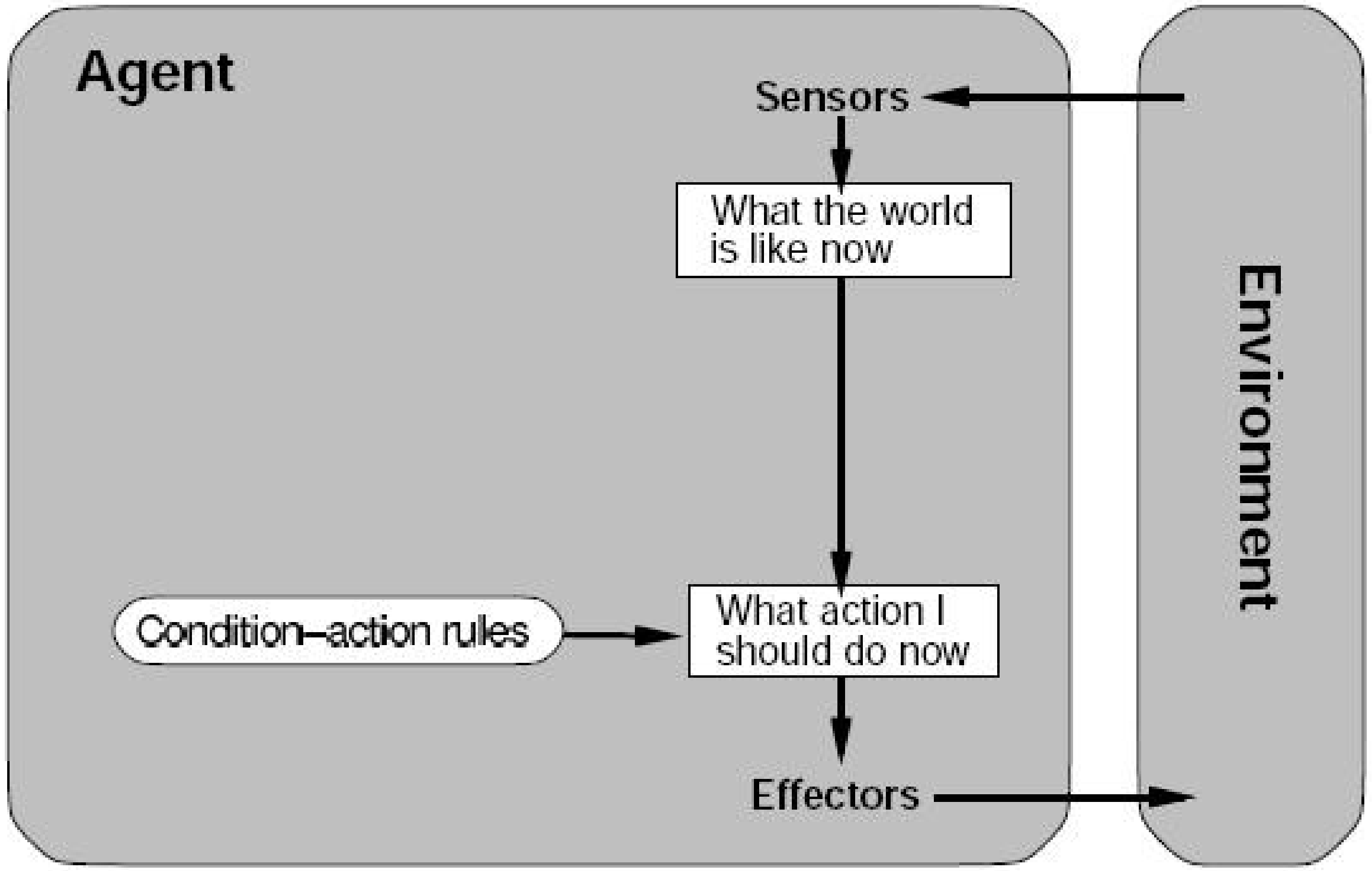
c. Goal-based agents

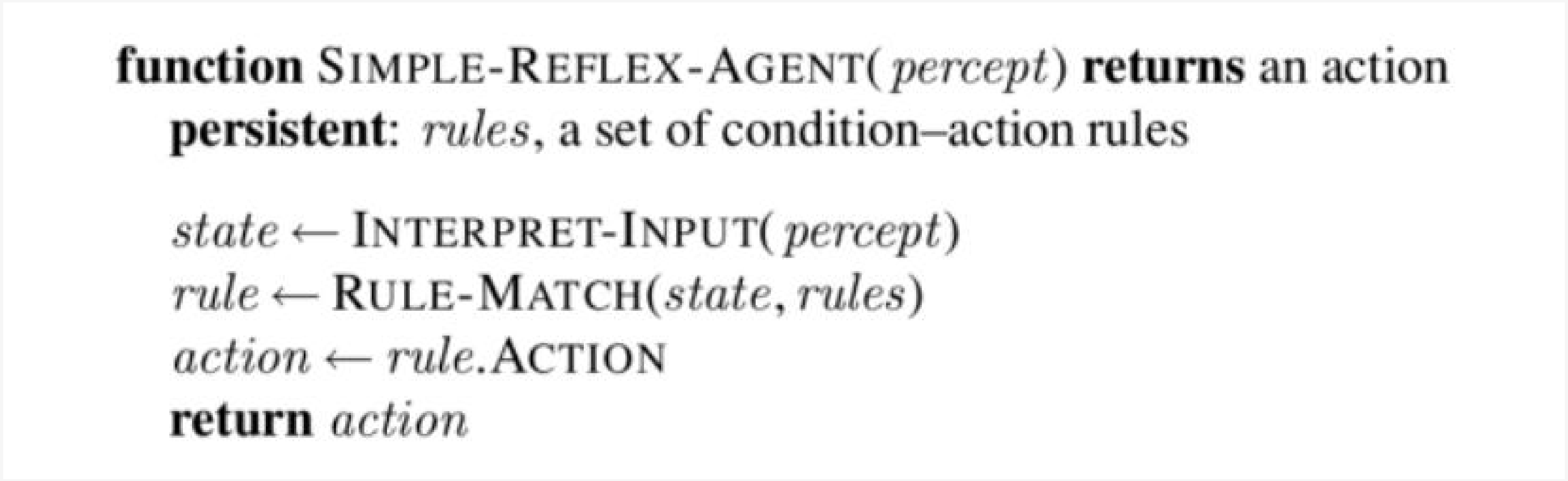
d. Utility-based agents

Each kind of agent program combines particular components in particular ways to generate actions.

If we convert each of these agents into learning agents then it can improve the performance of their components so as to generate better actions.

Simple Reflex Agents





```
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
```

Simple Reflex Agents

This agent selects actions based on the agents current perception or the world and not based on past perceptions.

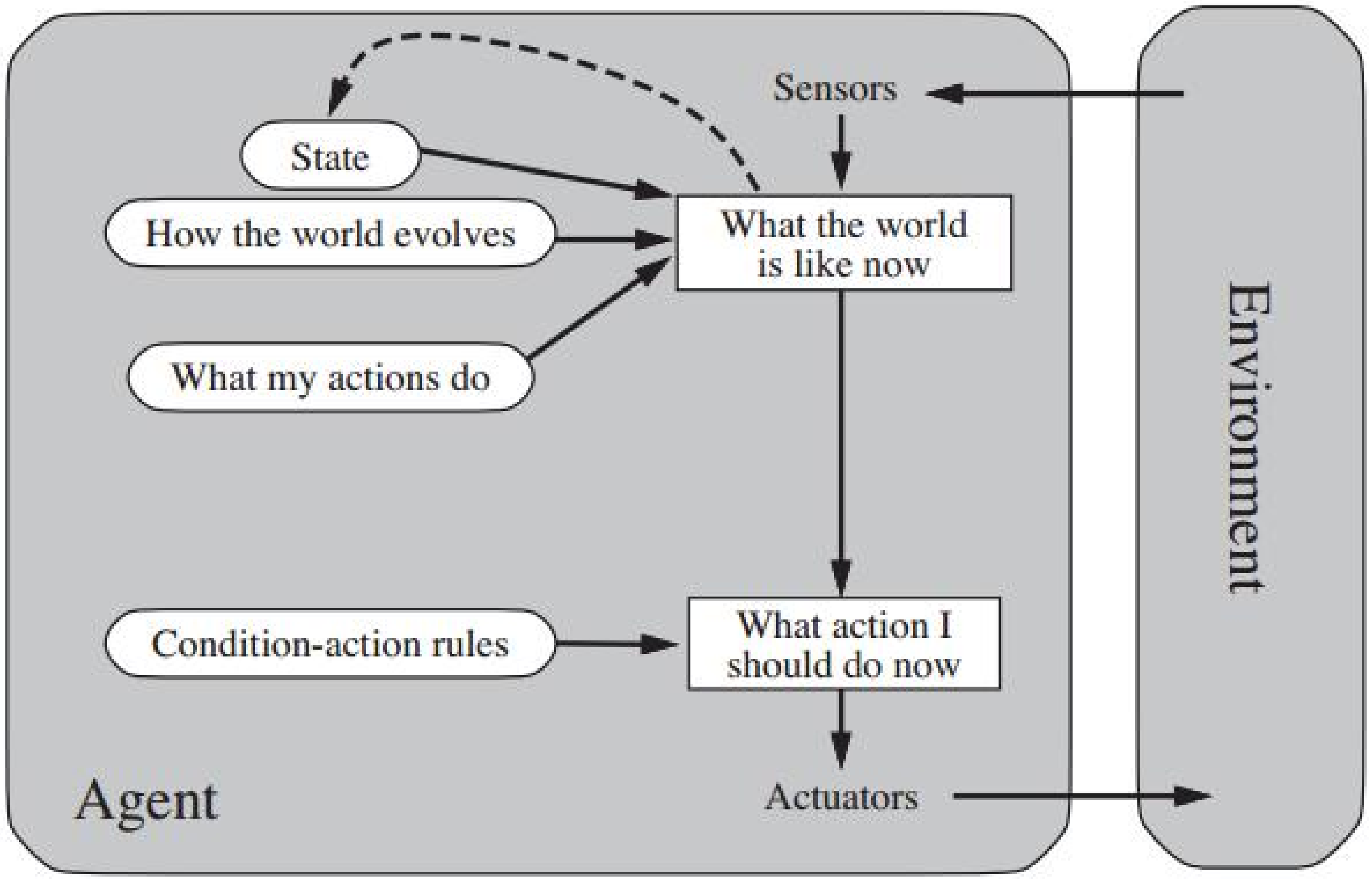
For example if a mars lander found a rock in a specific place it needed to collect then it would collect it, if it was a simple reflex agent then if it found the same rock in a different place it would still pick it up as it doesn't take into account that it already picked it up.

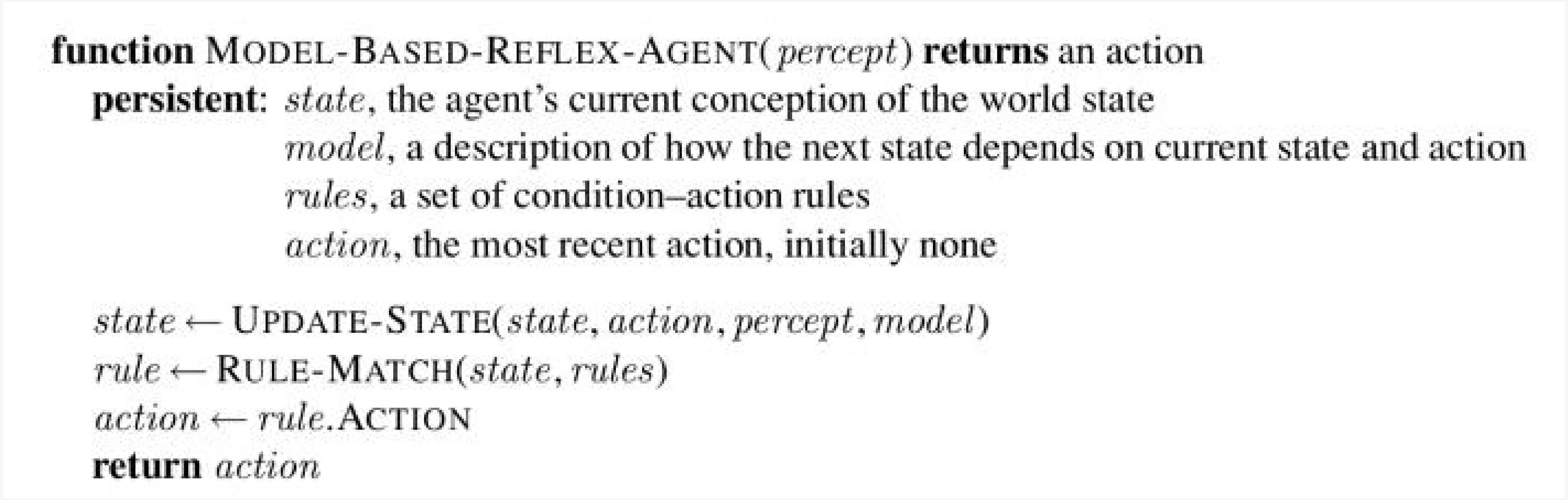
This is useful for when a quick automated response is needed, humans have a very similar reaction to fire for example, our brain pulls our hand away without thinking about any possibility that there could be danger in the path of your arm. We call these reflex actions.

This kind of connection where only one possibility is acted upon is called a condition action rule, written as:

if hand is in fire **then** pull away hand

Model-based Reflex Agents





```
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
```

Model-based Reflex Agents

Model-based Reflex Agents

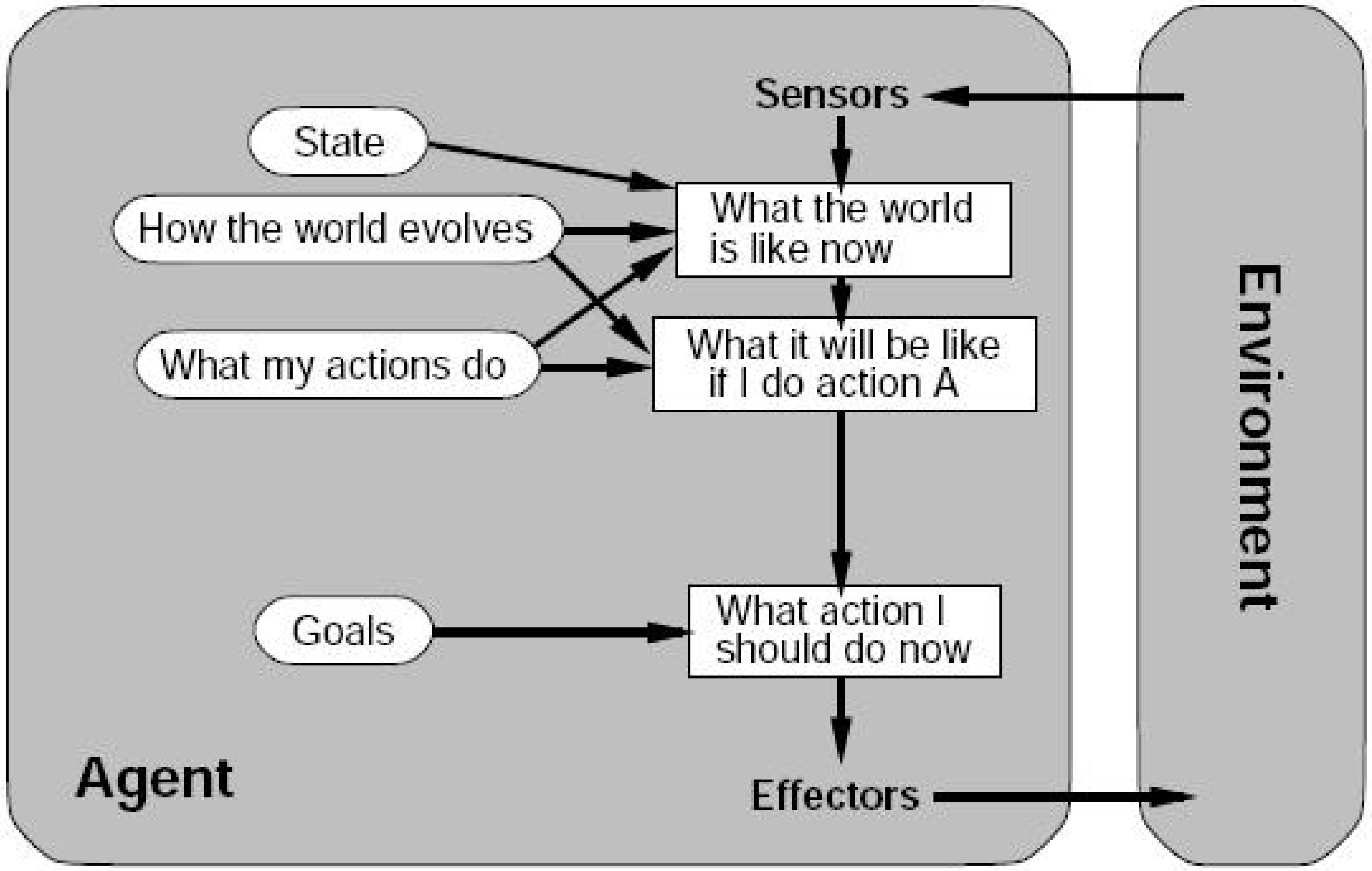
Model-based reflex agents are made to deal with partial accessibility; they do this by keeping track of the part of the world it can see now.

It does this by keeping an internal state that depends on what it has seen before so it holds information on the unobserved aspects of the current state.

This time out mars Lander after picking up its first sample, it stores this in the internal state of the world around it so when it come across the second same sample it passes it by and saves space for other samples.

While reading this you are keeping track of where you have got to somewhere internally in your brain just in case you lose your place.

Goal-based Agents


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
Effectors
Environment

In life, in order to get things done we set goals for us to achieve, this pushes us to make the right decisions when we need to.

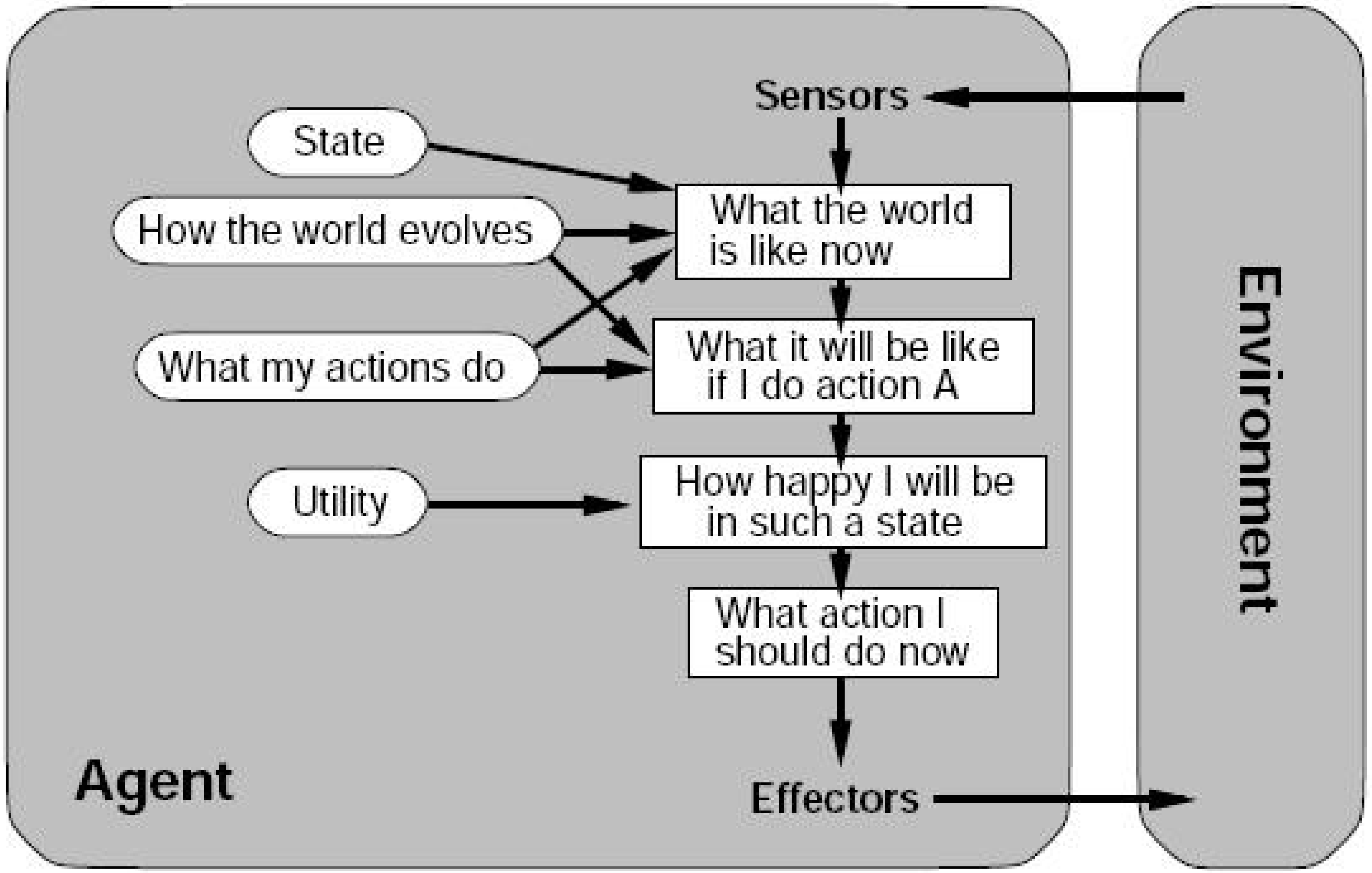
A simple example would be the shopping list; our goal is to pick up everything on that list. This makes it easier to decide if you need to choose between milk and orange juice because you can only afford one. As milk is a goal on our shopping list and the orange juice is not we chose the milk.

So in an intelligent agent having a set of goals with desirable situations are needed. The agent can use these goals with a set of actions and their predicted outcomes to see which action(s) achieve our goal(s).

Achieving the goals can take 1 action or many actions. Search and planning are two subfields in AI devoted to finding sequences of actions to achieve an agents goals.

Unlike the previous reflex agents before acting this agent reviews many actions and chooses the one which come closest to achieving its goals, whereas the reflex agents just have an automated response for certain

Utility-based Agents



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

Utility-based Agents

Just having goals isn't good enough because often we may have several actions which all satisfy our goal so we need some way of working out the most efficient one.

A utility function maps each state after each action to a real number representing how efficiently each action achieves the goal.

This is useful when we either have many actions all solving the same goal or when we have many goals that can be satisfied and we need to choose an action to perform.

For example let's show our mars Lander on the surface of mars with an obstacle in its way. In a goal based agent it is uncertain which path will be taken by the agent and some are clearly not as efficient as others but in a utility based agent the best path will have the best output from the

Learning Agents

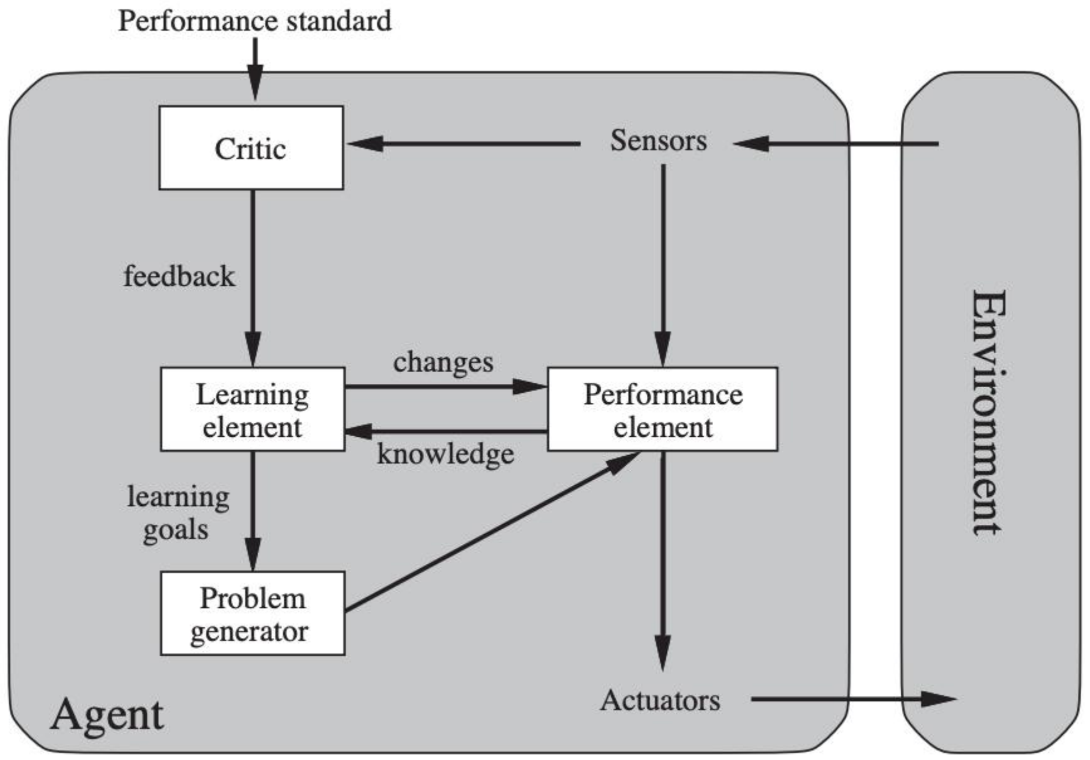


Figure 2.15 A general learning agent.

Learning Agents

When we expand our environments we get a larger and larger amount of tasks, eventually we are going to have a very large number of actions to pre-define.

Another way of going about creating an agent is to get it to learn new actions as it goes about its business, this still requires some initial knowledge but cuts down on the programming greatly.

This will allow the agent to work in environments that are unknown.

The learning element is responsible for improvements this can make a change to any of the knowledge components in the agents.

One way of learning is to observe pairs of successive states in the percept sequence; from this the agent can learn how the world evolves.

Learning Agents

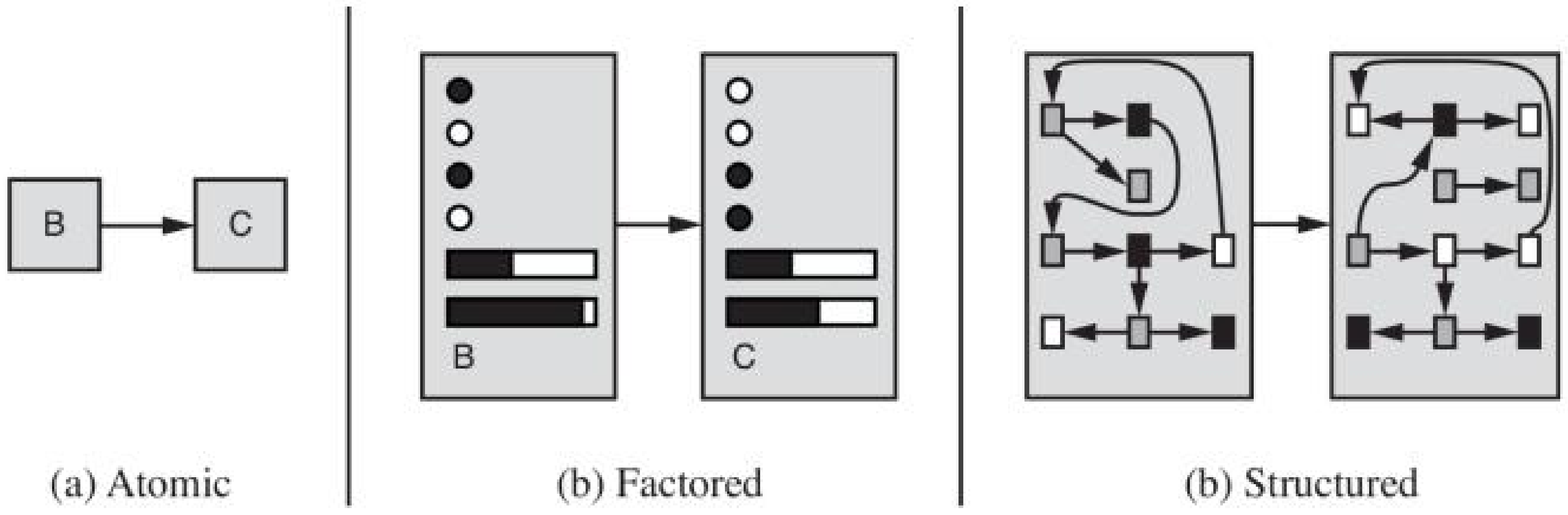
Learning element: It is responsible for making improvements by learning from the environment

Critic: The learning element takes feedback from critics which describes how well the agent is doing with respect to a fixed performance standard.

The learning agent gains feedback from the critic on how well the agent is doing and determines how the performance element should be modified if at all to improve the agent.

Performance element: It is responsible for selecting external action.

Problem Generator: This component is responsible for suggesting actions that will lead to new and informative experiences.



How the components of agent programs work?

Thank you