國立成功大學
National Cheng Kung University

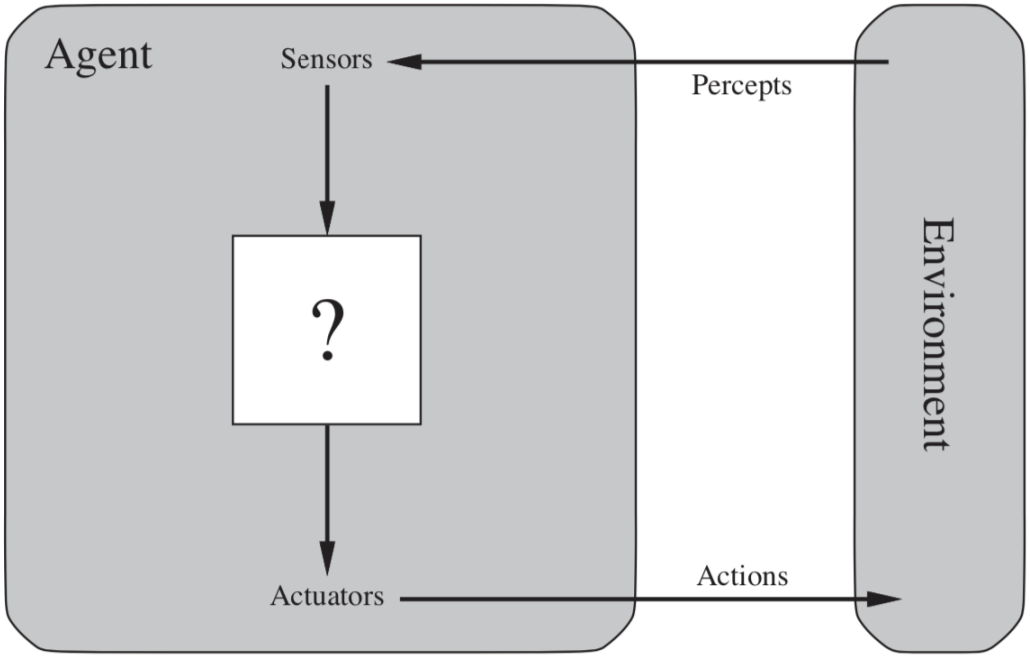
Introduction to Artificial Intelligence

Chapter 2
Intelligent Agents

Wei-Ta Chu (朱威達)

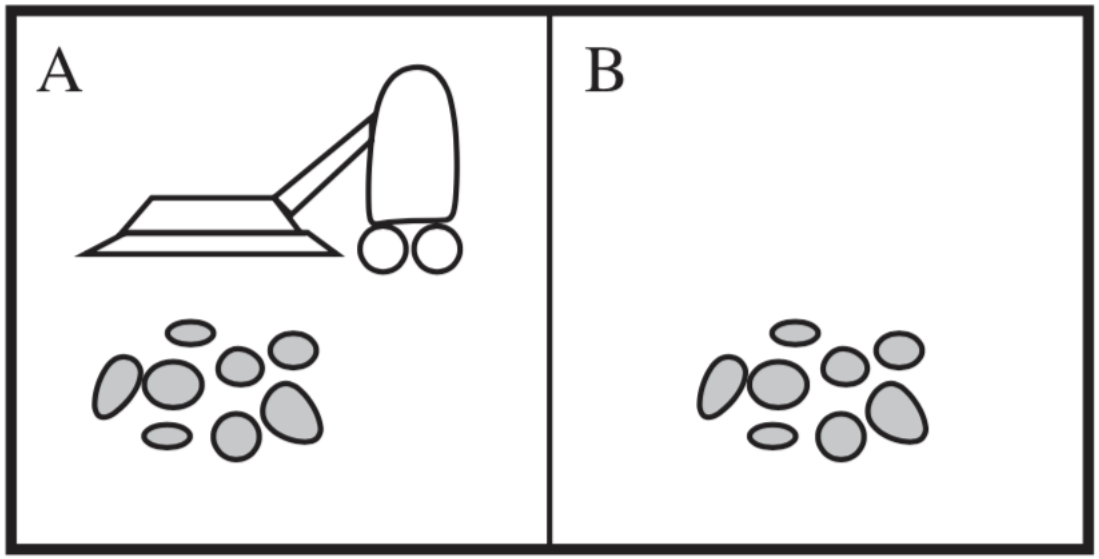
Agents and Environments

- An **agent** is anything that can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators (促動器)**.

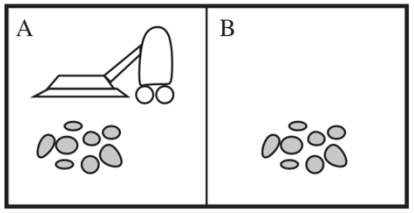


Agents and Environments

- The term **percept** refers to the agent's perceptual inputs at any given instant.
- An agent's behavior is described by the **agent function** that maps any given percept sequence to an action.
- A very simple example—the vacuum-cleaner world



Agents and Environments



- It can choose to move left, move right, suck up the dirt, or do nothing. One very simple agent function: if the current square is dirty, then suck; otherwise, move to the other square.
- What is the right way to fill out the table? What makes an agent good or bad, intelligent or stupid?

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.

The Concept of Rationality

- A **rational agent** is one that does the right thing.
- Right thing: The agent's actions causes the environment to go through a sequence of states. If the sequence is desirable, then the agent has performed well. This notion of desirability is captured by a **performance measure** that evaluates any given sequence of environment states.
- It is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave.

Rationality

- Definition of a **rational agent**
  - For 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 percept sequence and whatever built-in knowledge the agent has.
- Consider the simple vacuum-cleaner agent. Is this a rational agent? That depends!
  - Performance measure: the amount of dirt being cleaned up
  - Performance measure: a clean floor

Omniscience, Learning, and Autonomy

- Rationality maximizes **expected** performance, while perfection maximizes **actual** performance.
- Our definition of rationality does not require omniscience (全知), because the rational choice depends only on the percept sequence to date.
- Our definition requires a rational agent not only to **gather information** but also to **learn** as much as possible from what it perceives.

Omniscience, Learning, and Autonomy

- A rational agent should be **autonomous**—it should learn what it can to compensate for partial or incorrect prior knowledge.
  - A vacuum-cleaning agent that learns to foresee where and when additional dirt will appear will do better than one that does not.
- The incorporation of **learning** allows one to design a single rational agent that will succeed in a vast variety of environments.

Task Environments

- Task environment: we have to specify the performance measure, the environment, and the agent's actuators and sensors.
  - **PEAS** (Performance, Environment, Actuators, Sensors)

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.

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.

Properties of Task Environments

- Fully observable vs. partially observable
  - Fully observable: if the sensors detect all aspects that are relevant to the choice of action
- Single agent vs. multiagent
  - The key distinction is whether B's behavior is best described as maximizing a performance measure whose value depends on agent A's behavior.
  - Chess is a competitive multiagent environment

Properties of Task Environments

- Deterministic vs. stochastic
  - If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is *deterministic*; otherwise, it is *stochastic*.
  - Taxi driving is clearly stochastic
  - "Stochastic" generally implies that uncertainty about outcomes is quantified in terms of probabilities

Properties of Task Environments

- Episodic (情節不連貫的) vs. sequential
  - Episodic: the next episode does not depend on the actions taken in previous episodes.
  - Chess and taxi driving are sequential
- Static vs. dynamic
  - If the environment can change while an agent is deliberating (思考), then we say the environment is dynamic for that agent.
  - Taxi driving is clearly dynamic: the other cars and the taxi itself keep moving while the driving algorithm dithers (猶豫) about what to do next.

Properties of Task Environments

- Discrete vs. continuous
  - The chess environment has a finite number of distinct states. Chess also has a discrete set of percepts and actions. Taxi driving is a continuous-state and continuous-time problem.
- Known vs. unknown
  - The agent's (or designer's) state of knowledge about the "laws of physics" of the environment.

Properties of Task Environments

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

- The job of AI is to design an **agent program** that implements the agent function— the mapping from percepts to actions.
- We assume this program will run on some sort of computing device with physical sensors and actuators—we call this the **architecture**.
- **agent = architecture + program**

Agent Programs

- A trivial agent program

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.

- The table-driven approach to agent construction is doomed to failure – tables could be too huge

Simple Reflex Agents

- Select actions on the basis of the current percept, ignoring the rest of the percept history.

```
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
  - Work only if the correct decision can be made on the basis of only the current percept—that is, only if the environment is fully observable.

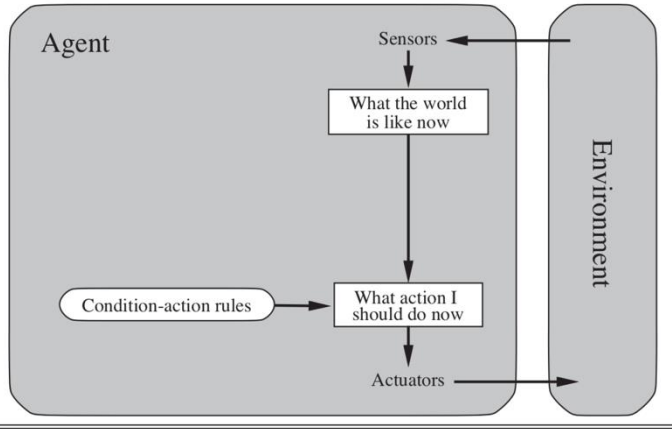


Figure 2.9 Schematic diagram of a simple reflex agent.

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.

Model-based Reflex Agents

- Handle partial observability: the agent keeps track of the part of the world it can't see now. The agent should maintain some sort of **internal state** that depends on the percept history.

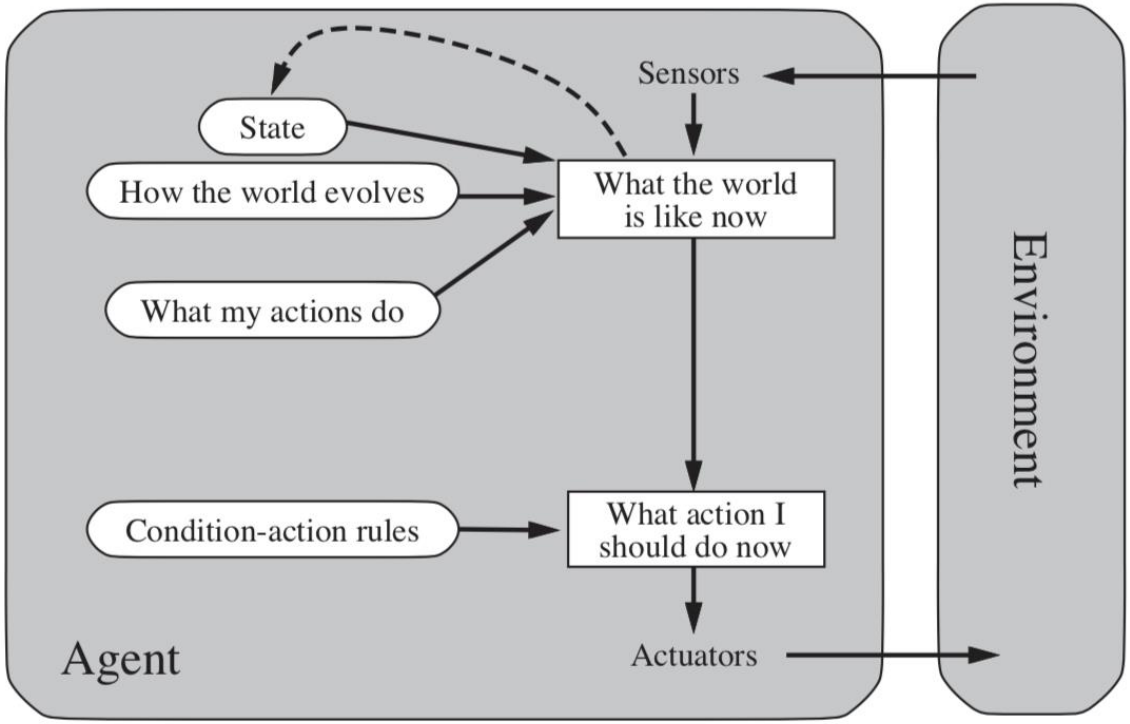


Figure 2.11 A model-based reflex agent.

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*

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 Reflex Agents

- Knowing the current state of environment is not always enough to decide what to do. The agent needs some sort of **goal** information that describes situations that are desirable.
- The goal-based agent's behavior can easily be changed to go to a different destination, simply by specifying that destination as the goal.
- Decision making different from condition-action rules
  - Consideration of the future – both "What will happen if I do such-and-such?" and "Will that make me happy?"

Goal-based Reflex Agents

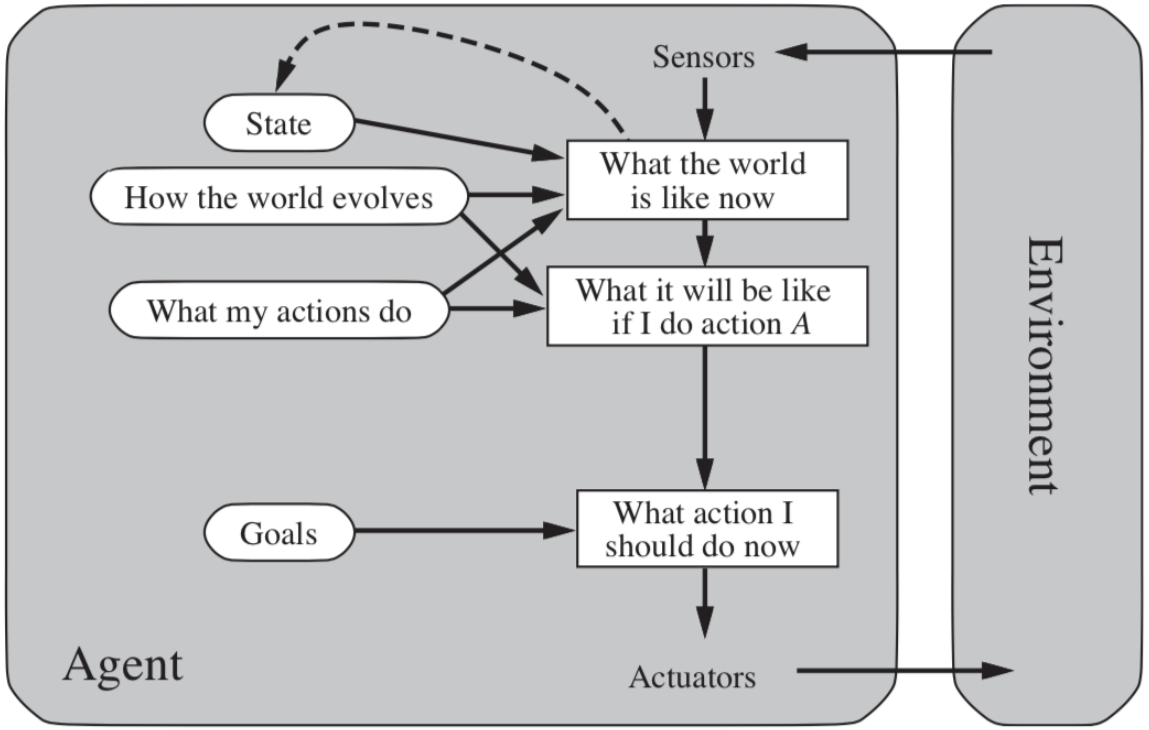


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Utility-based Reflex Agents

- Goals alone are not enough to generate high-quality behavior in most environments. Sometimes, goals are inadequate but a utility-based agent can still make rational decisions.
  - When there are conflicting goals, only some of which can be achieved
  - When there are several goals that the agent can aim for
- A rational utility-based agent chooses the action that maximizes the expected utility of the action outcomes.

Utility-based Reflex Agents

- Utility-based agent programs handle the uncertainty inherent in stochastic or partially observable environments.

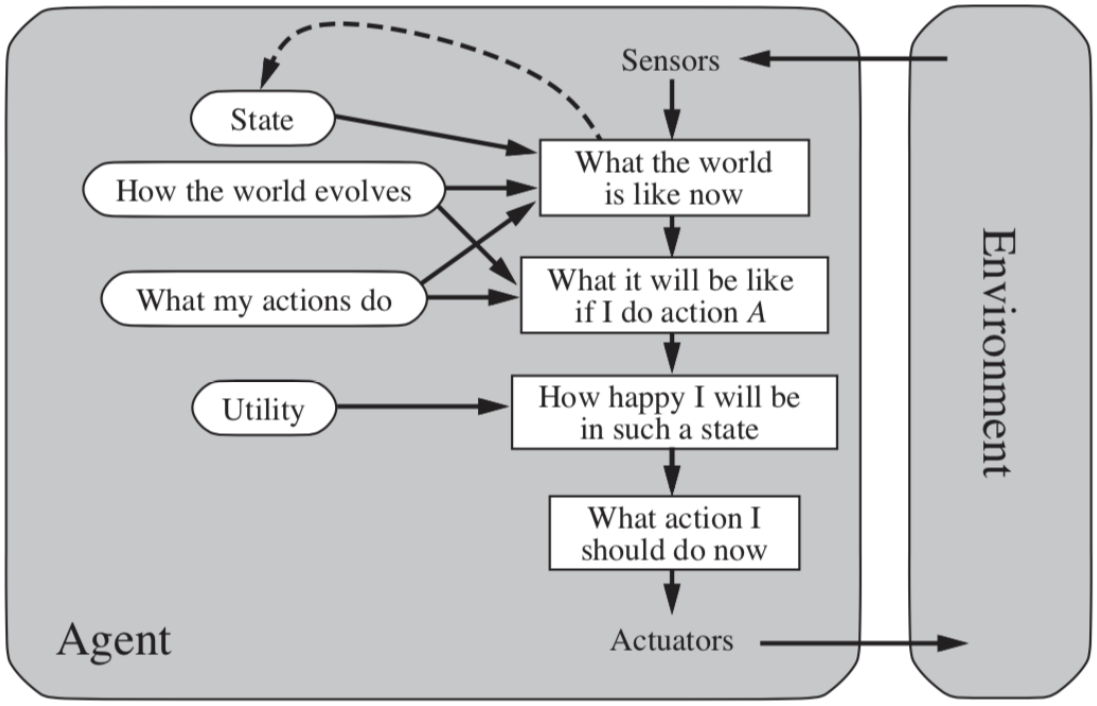


Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome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

Learning Agents

- **Learning element** is responsible for making improvements, and the **performance element** is responsible for selecting external actions.
- The learning element uses feedback from the **critic** on how the agent is doing and determines how the performance component should be modified to do better in the future.

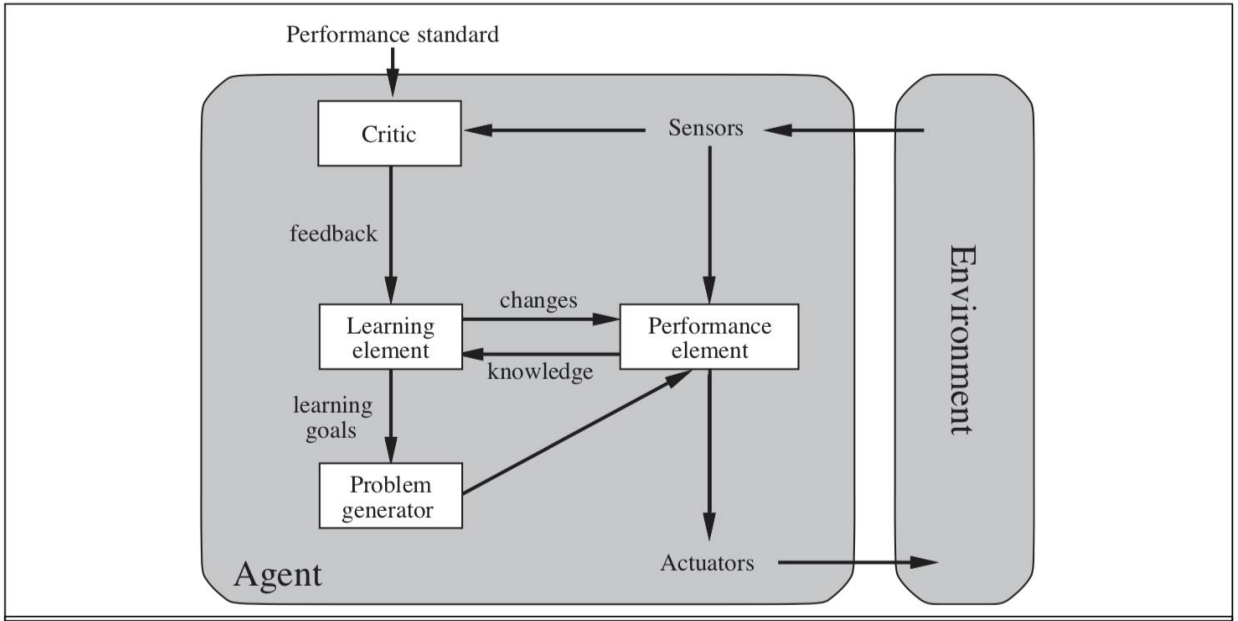


Figure 2.15 A general learning agent.

Learning Agents

- **Problem generator** is responsible for suggesting actions that will lead to new and informative experiences.
- If the agent is willing to explore a little and do some perhaps suboptimal actions in the short run, it might discover much better actions for the long run. The problem generator's job is to suggest these exploratory actions.
- Learning in intelligent agents can be summarized as a process of modification of each component of the agent to bring the components into closer agreement with the available feedback information.