Unit 1 Chapter 2

Intelligent Agents

Agents and environments, good behavior, concept of rationality, nature of environments, structure of agents.

Review of Intelligent Agents

- Motivation
- Objectives
- Introduction
- Agents and Environments
- Rationality
- Agent Structure
- Agent Types
  - Simple reflex agent
  - Model-based reflex agent
  - Goal-based agent
  - Utility-based agent
  - Learning agent

Introduction to Intelligent Agents

- Motivation
  - Agents are used to provide a consistent viewpoint on various topics in AI
  - Agents require essential skills to perform tasks that require intelligence
  - Intelligent agents use methods and techniques from the field of AI
- What is an agent?
  - In general, an entity that interacts with its environment
    - Perception through sensors
    - Actions through effectors or actuators
- Agent and its environment
  - An agent perceives its environment through sensors
    - The complete set of inputs at a given time is called a percept
    - The current percept, or a sequence of percepts may influence the actions of an agent
  - It can change the environment through actuators
    - An operation involving an actuator is called an action
    - Actions can be grouped into action sequences

AGENTS and ENVIRORNMENTS

- An **agent** is anything that can be ENVIRONMENT viewed as perceiving its **environment** through **sensors** and SENSOR acting upon that environment through **actuators**.
- **ACTUATOR** :A human agent has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators.
- **A robotic agent** : 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.

Intelligent Agents

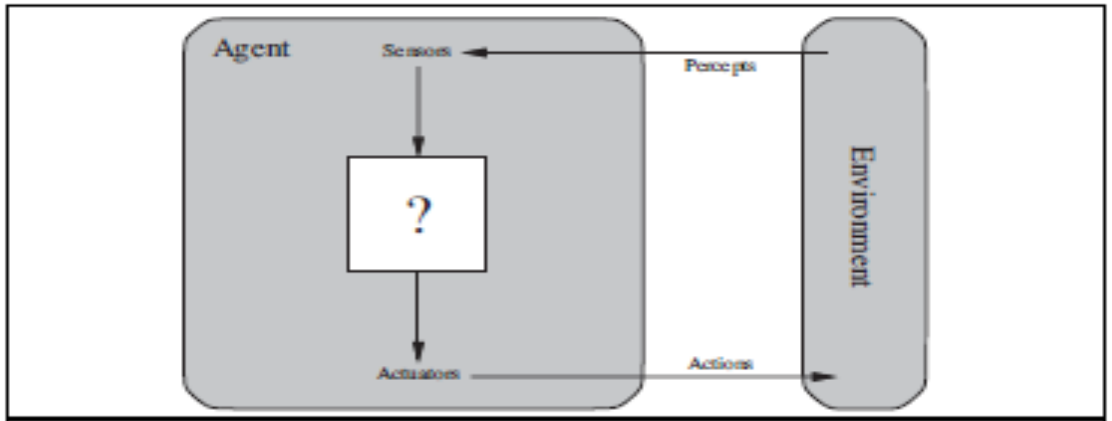
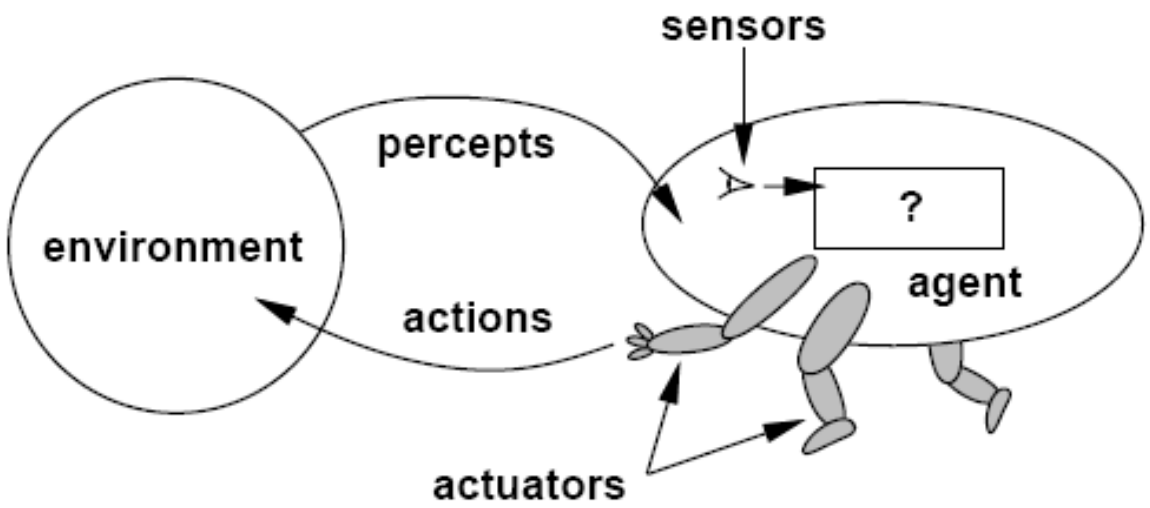


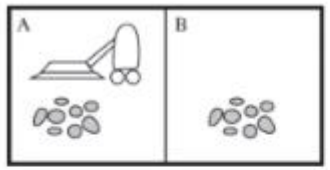
Figure 2.1 Agents interact with environments through sensors and actuators.



Basic Concepts

- **percept :** agent's perceptual inputs at any given instant.
- agent's **percept sequence :** complete history of everything the agent has ever perceived.
- *agent's choice of action : entire percept sequence observed to date, but not on anything it hasn't perceived.*
- agent's behavior is :**agent function** that maps any given percept sequence to an action.
- *tabulating* the agent function that describes any given agent;
- *Internally*, the agent function for an artificial agent will be implemented by an **agent program**.
- The agent function is an abstract mathematical description

Vacuum-Cleaner World



- Only two locations: A, B
- Floor clean or dirty
- Percept: [A, clean]
- Actions: right, left, suck

Simple agent function:

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], [B, clean]	left
[A, clean], [B, dirty]	suck
[A, dirty], [A, clean]	right
…	

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

GOOD BEHAVIOR: THE CONCEPT OF RATIONALITY

- rational agent does “the right thing”
  - The action that leads to the best outcome under the given circumstances
- An agent function maps percept sequences to actions
  - Abstract mathematical description
- An agent program is a concrete implementation of the respective function
  - It runs on a specific agent architecture (“platform”) on physical devices.
- Problems:
  - What is “ the right thing”
  - How do you measure the “best outcome”

Rationality

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.

- **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.

Rational agent and Omniscience

- **Omniscience**
- An omniscient agent knows the *actual* outcome of its actions and can act accordingly; but omniscience is impossible in reality
- **learning**
- Rational agent:
- Learn and gather information from what it perceives.
- The agent's initial configuration could reflect some prior knowledge of the environment, but as the agent gains experience this may be modified and augmented.
- There are extreme cases in which the environment is completely known *a* ***priori*****.**

Rational agent and Omniscience

- **autonomy**
- A rational agent should be autonomous—it should learn what it can to compensate for partial or incorrect prior knowledge.
- **task environment**
- Task environments, which are essentially the “problems” to which rational agents are the “solutions.

Specifying the task environment

- **PEAS** (**P**erformance, **E**nvironment, **A**ctuators, **S**ensors)
- An automated taxi driver.

Task environment

- The full driving task is extremely ***open-ended.***
- PEAS description for the taxi's 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.

- **software agents** (or software robots or **softbots**) exist in rich, unlimited domains.
- Imagine a softbot Web site operator designed to scan Internet news sources and show the interesting items to its users, while selling advertising space to generate revenue.

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.

Properties of task environments

- **Fully observable** vs. **partially observable**:
- **Single agent** vs. **multiagent**
- **Deterministic** vs. **stochastic**
- **Episodic** vs. **sequential**
- **Static** vs. **dynamic**:
- **Discrete** vs. **continuous**
- **Known** vs. **unknown**

Properties of task environments

1. **Fully observable** vs. **partially observable**:

- If an agent's sensors give it access to the complete state of the environment at each point in time.
- All aspects that are *relevant* to the choice of action;
- agent need not maintain any internal state
- partially observable: because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data.
- a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares, and an automated taxi cannot see what other drivers are thinking.
- If the agent has no sensors at all then the environment is **unobservable**.

Properties of task environments

2.Single agent vs. multiagent:

- an agent solving a crossword puzzle by itself is clearly in a single-agent environment
- agent playing chess is in a two agent environment.

3.Deterministic vs. stochastic

- If the next state of the environment is completely determined by the current state and the action executed by the agent.
- otherwise, it is stochastic.
- If partially observable-→ could be stochastic.
- The environment is uncertain if it is not fully observable or not deterministic.
- Nondeterministic environment is one in which actions are characterized by their possible outcomes, but no probabilities are attached to them.

4.Episodic vs. sequential

- The agent’s experience is divided into atomic episodes.
- In each episode the agent receives a percept and then performs a single action.
- Next episode does not depend on the actions taken in previous episodes. Many classification tasks are episodic
- **sequential** :
- current decision could affect all future decisions.
- Ex:Chess and taxi driving are sequential: in both cases, short-term actions can have long-term consequences.
- Episodic environments are **much simpler** than sequential environments because the agent does not need to think ahead.

5.**Static** vs. **dynamic**

- The environment and agent changes - dynamic otherwise, static.
- Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time.
- Ex:Crossword puzzles are static.
- Dynamic environments are continuously asking the agent what it wants to do; if it hasn’t decided yet, that counts as deciding to do nothing.
- Ex: Taxi driving
- If the **environment** itself does not change with the passage of time but the agent’s performance score does – **semidynamic.**
- **Ex:**Chess, when played with a clock, is semidynamic

6. Discrete vs. **continuous**:

- *state* of the environment, to the way *time* is handled, and to the *percepts* and *actions* of the agent.
- For example, the chess environment has a finite number of distinct states (excluding the clock). Chess also has a discrete set of percepts and actions.
- Taxi driving is a continuous-state and continuous-time problem: the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and do so smoothly over time.

7.Known vs. unknown:

- Known: **outcomes or outcome probabilities for all actions** are given.
- Unknown: the agent will have to learn how it works in order to make good decisions.
- It is quite possible for a ***known*** **environment to be *partially* observable**—
- Ex: solitaire card games: I know the rules but am still unable to see the cards that have not yet been turned over.
- *Unknown* environment: can be *fully* observable—in a new video game, the screen may show the entire game state but I still don't know what the buttons do until I try them.

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.
- program will run on computing device with physical sensors and actuators—the **architecture**:
- *agent = architecture + program*

THE STRUCTURE OF AGENTS

- **Agent programs**
- The agent program takes just the current percept as input;
- If the agent's actions need to depend on the entire percept sequence, the agent will have to remember the percepts.

```
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.

- **Agent programs**
- Let **P** be the set of possible percepts and let T be the lifetime of the agent (the total number of percepts it will receive).
- The lookup table will contain entries.

$$\sum_{t=1}^{T} |\mathcal{P}|^t$$

- Ex: Consider the automated taxi: the visual input from a single camera comes in at therate of roughly 27 megabytes per second (30 frames per second, 640×480 pixels with 24bits of color information). This gives a lookup table with over $10^{250,000,000,000}$ entries for an hour's driving.

THE STRUCTURE OF AGENTS

- **Agent programs**
- size of these tables

a) Table entries are larger

b) Designer would not have time to create the table

c) All the right table entries cannot be learnt from its experience,

d) In a simple environment lack of guidance to fill the table entry is requirred .

- TABLE-DRIVEN-AGENT *does* do what we want: it implements the desired agent function.

- Simple reflex agents;
- Model-based reflex agents;
- Goal-based agents; and
- Utility-based agents.

Simple reflex agents

- simplest kind of agent.
- Select actions based **on *current* percept**, ignoring **percept history**

Ex: vacuum agent: its decision is based only on the current location and on whether that location contains dirt.

```
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.

- **condition–action rule: if** *car-in-front-is-braking* **then** *initiate-braking*

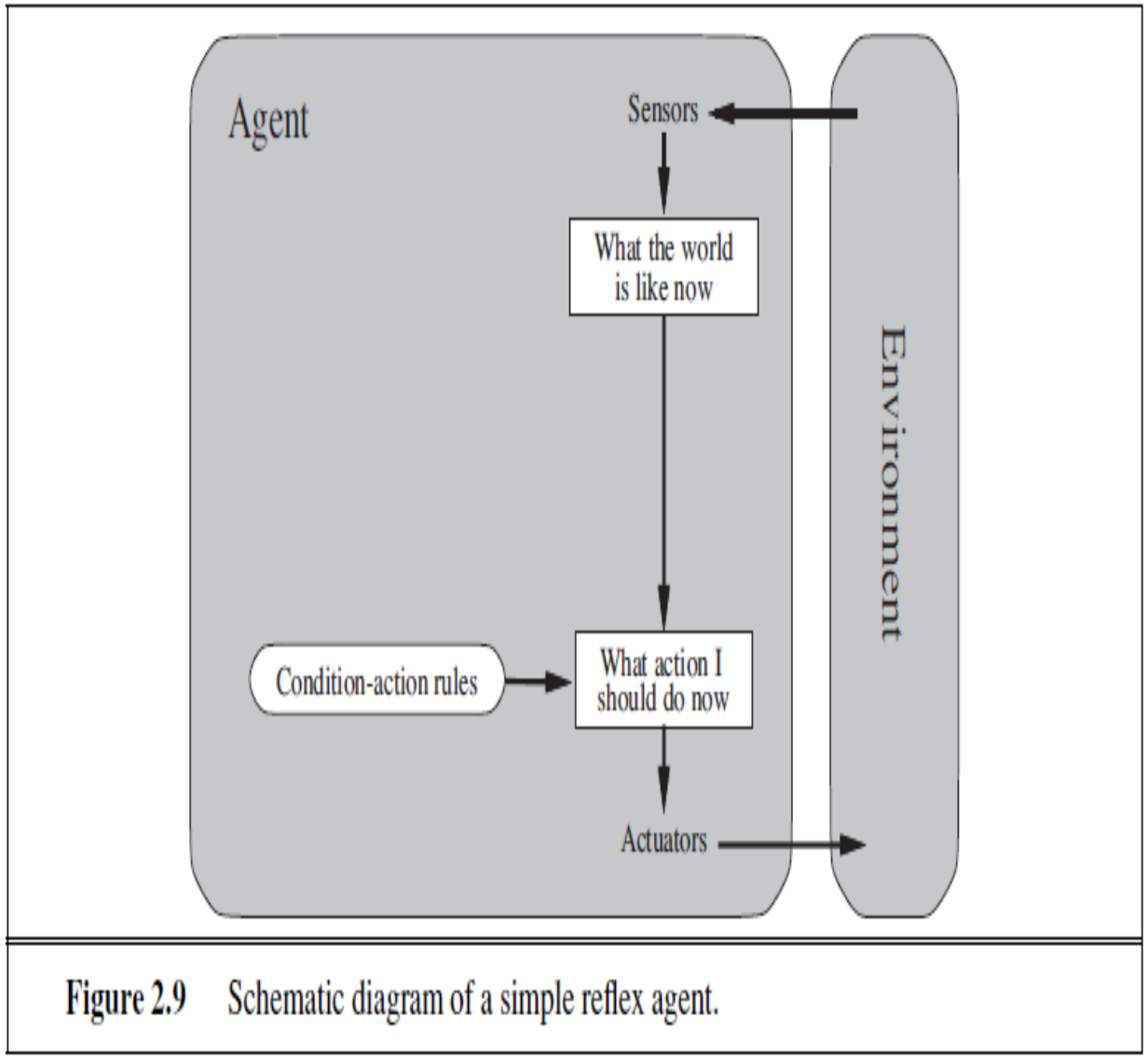


Figure 2.9 Schematic diagram of a simple reflex agent.

- Rectangles:current internal state of the agent's decision process
- Ovals:background information used in the process.

```
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.

actual implementations :using logic gates with a Boolean circuit.

Limitations:

1.Admirable property of being simple, but they turn out to be of limited intelligence.

Ex: *only the current percept—that is, only if the environment is fully observable.*

Example: Car Braking .

2.Escape from infinite loops is possible if the agent can randomize its actions.

Model-based reflex agents

- The solution to partial observability is to ***keep track of the part*** *of the world it can't see now*.
- Agent – Maintain **internal state with** percept history and thereby reflects at least some of the **unobserved aspects of the current state**.

Model-based reflex agents

- Updating the internal states
- **First, some information about how the world evolves independently of the agent**
- Ex: Overtaking car generally will be closer behind than it was a moment ago.
- **Second, need some information about how the agent's own actions affect the world**
- Ex: when the agent turns the steering wheel clockwise, the car turns to the right, or anticlockwise towards left.

Model-based reflex agents

- knowledge about "how the world work is called a **model** of the world. An agent that uses such a model is called a **model-based agent**.

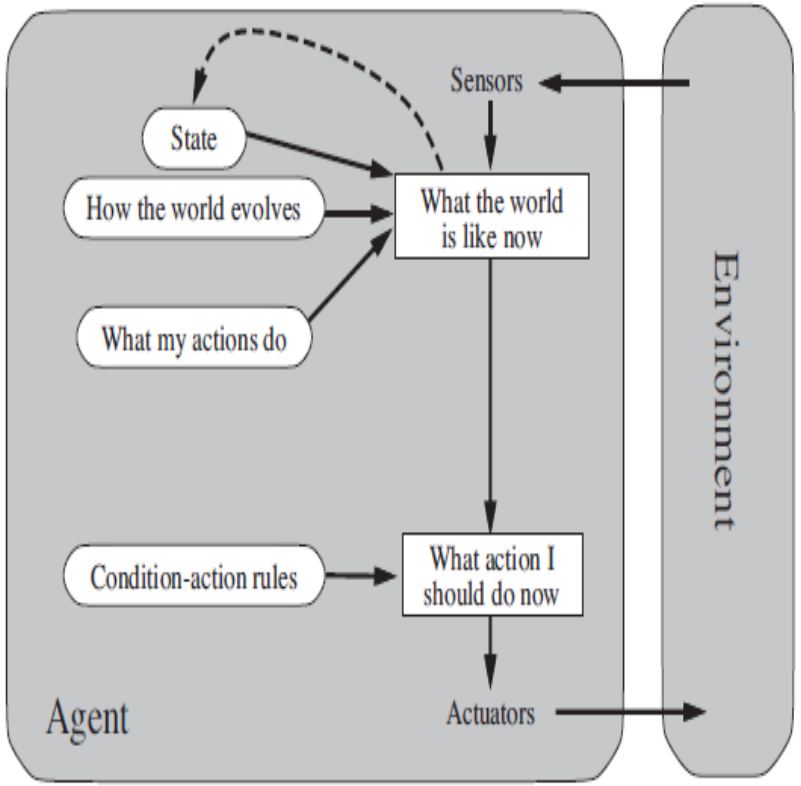


Figure 2.11 A model-based reflex agent.

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

- current state description, the GOAL agent needs some sort of **goal** information with situation desirable Ex: passenger's destination.
- The agent program can combine this with the model actions that achieve the goal.

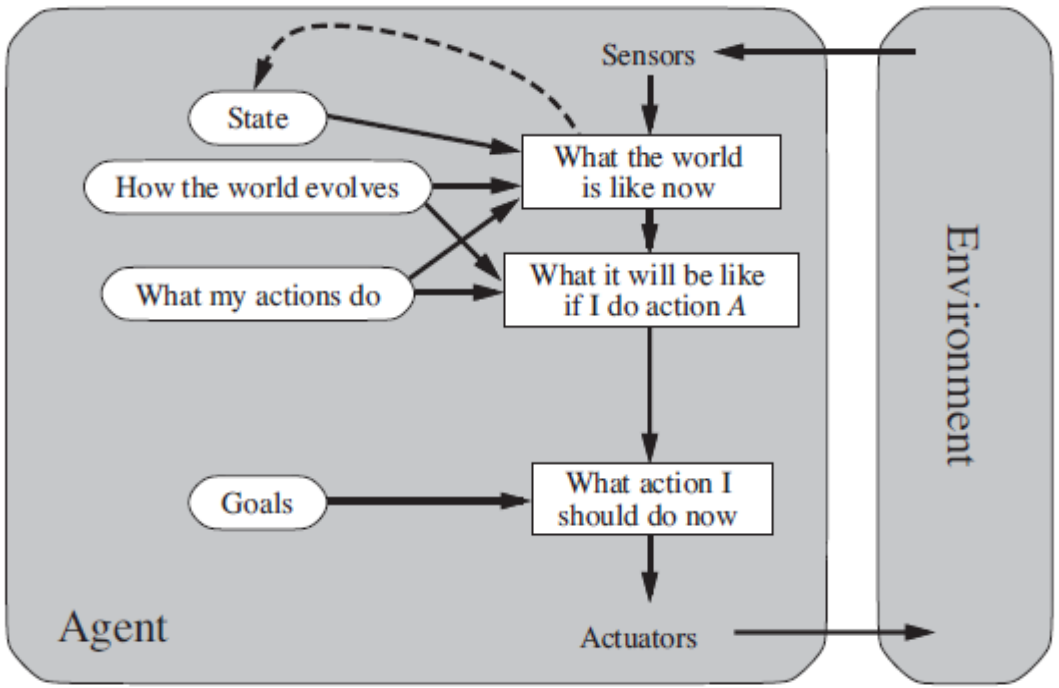


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Difference between goal based and reflex agents

Goal based :

- Less efficient, it is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified.
- The goal-based agent's behavior can easily be changed to go to a different destination, simply by specifying that destination as the goal

Reflex agent,

- Need to rewrite many **condition–action r**ules.
- The reflex agent's rules for when to turn and when to go straight will work only for a single destination; they must all be replaced to go somewhere new

Utility-based agents

❖Goals- “happy” and “unhappy” states.

- Because “happy” is not very scientific, economists and computer scientists use the term **utility** instead

❖An agent’s **utility function is** performance measure.

❖ If the internal utility function and the external performance measure are in agreement, then an agent that chooses actions to maximize its utility will be rational according to the external performance measure.

❖many advantages in terms of flexibility and learning.

Utility-based agents

- goals are inadequate but a utility-based agent can still make **rational decisions**.
- **First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff.**
- **Second, decision making under uncertainty**

Utility-based agents

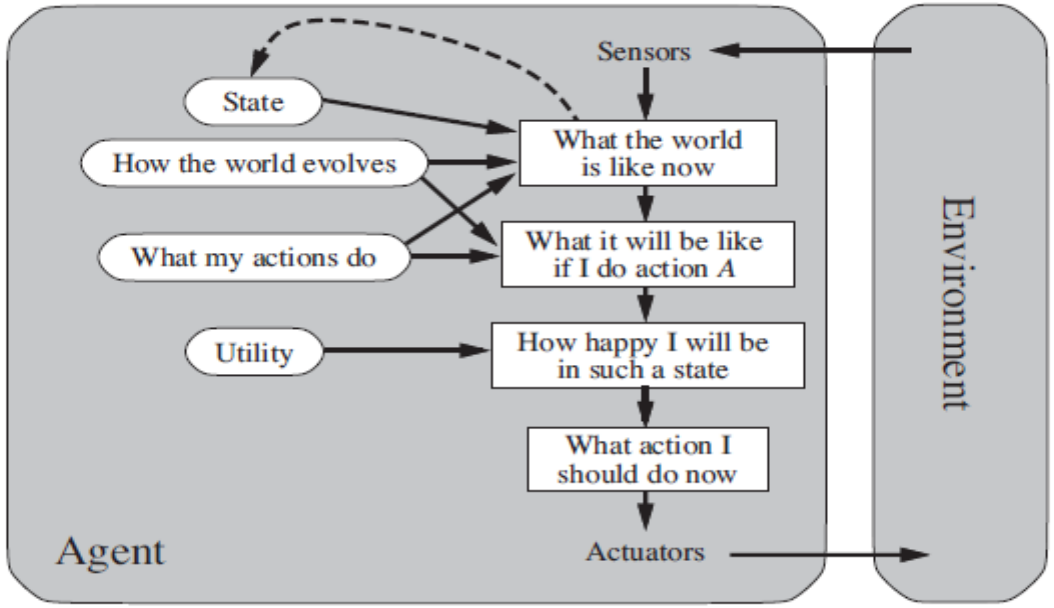


Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome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

Learning agents

- **how the agent programs *come into being***

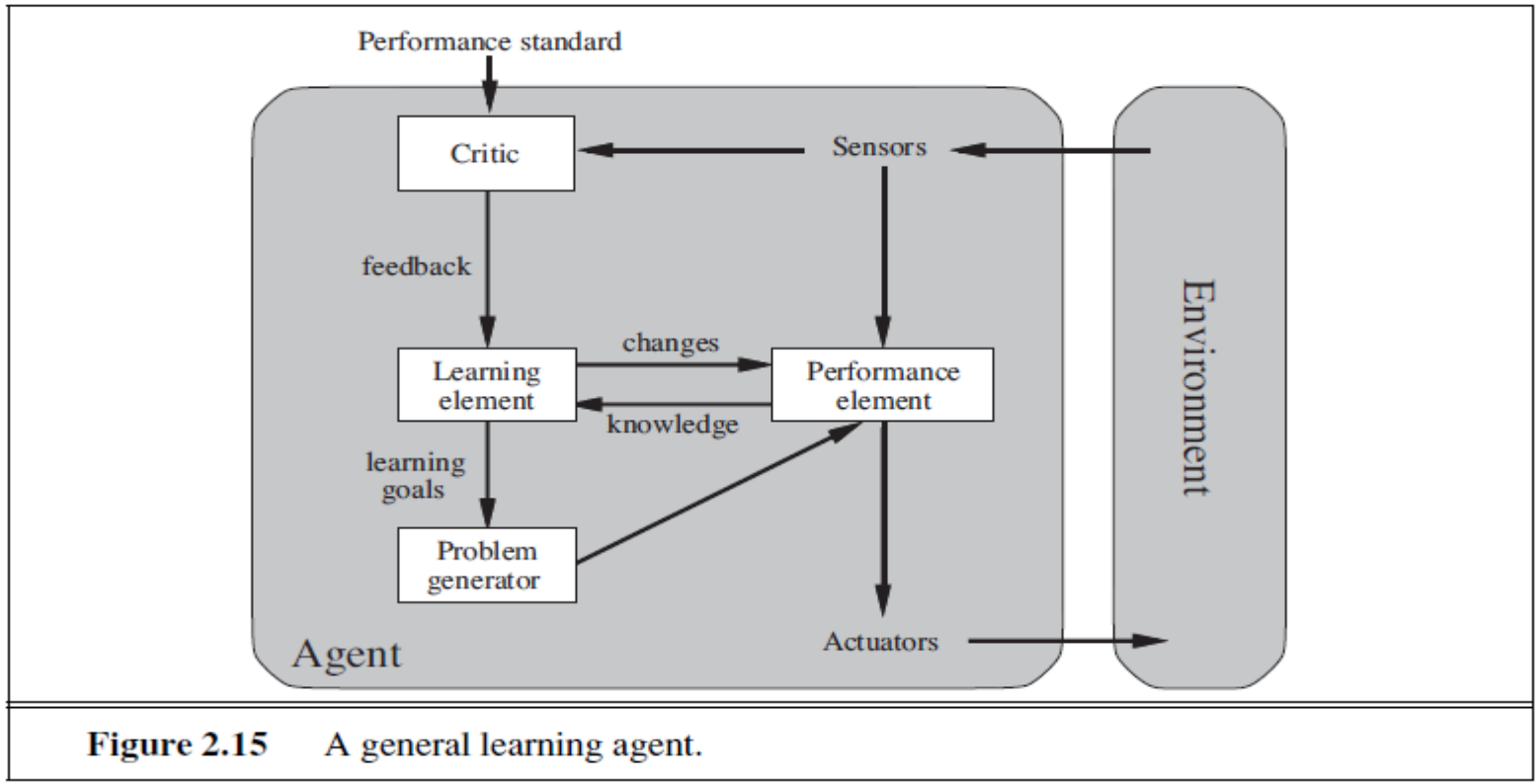


Figure 2.15 A general learning agent.

- the performance standard distinguishes part of the incoming percept as a **reward** (or **penalty**) that provides direct feedback on the quality of the agent’s behavior.

How the components of agent programs work

- What is the world like now?”
- “What action should I do now?”
- “What do my actions do?”
- next question for a student of AI How on earth do these components work?”
- representations along an axis of increasing complexity and expressive power—**atomic**, **factored**, and **structured**

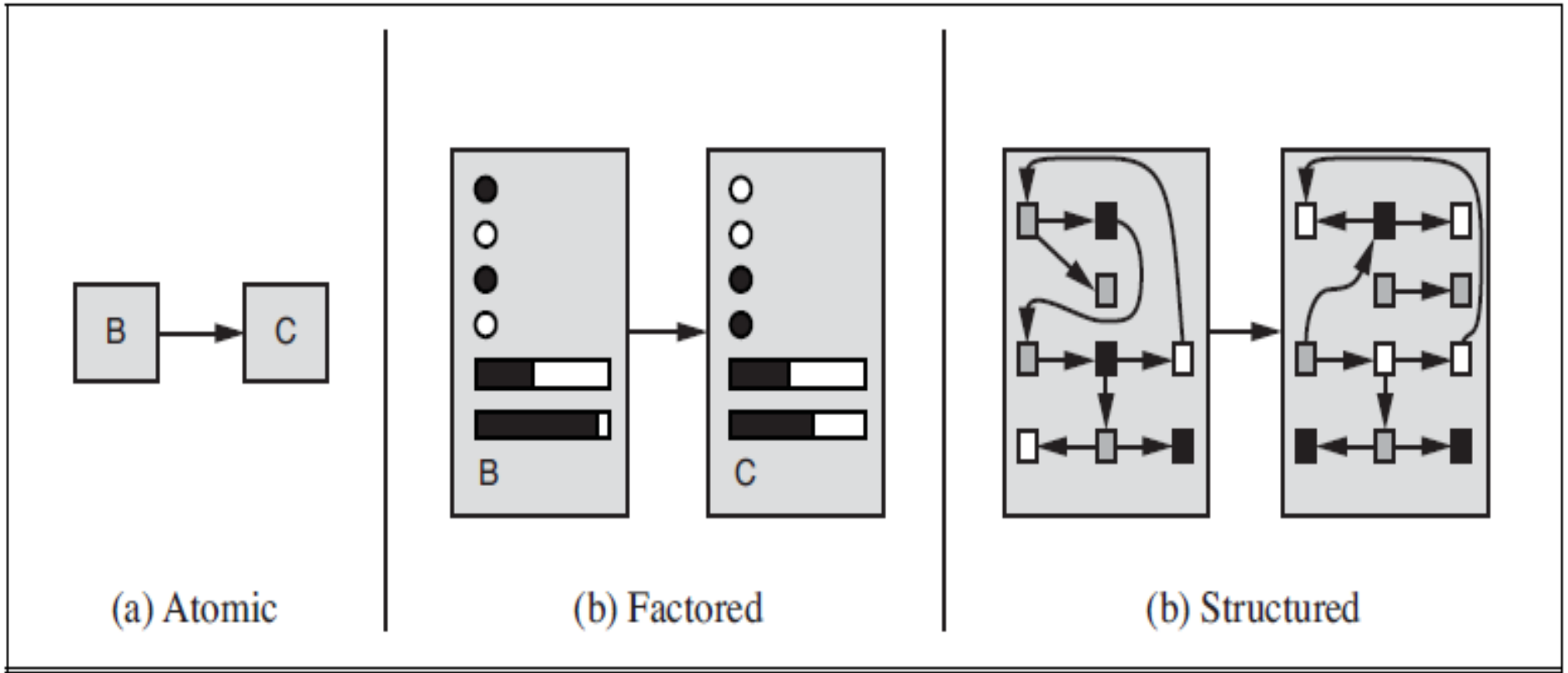


Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed 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.

How the components of agent programs work

- *atomic representation*

❑each state of the world is indivisible-it has no internal structure.

❑Ex:just atomic location in one city or another.

- factored representation

- splits up each state into a fixed set of variables or attributes, each of which can have a value.
- how much gas is in the tank, our current GPS coordinates, whether or not the oil warning light is working, how much spare change we have for toll crossings, what station is on the radio, and so on.

structured representation

- Ex: TruckAheadBackingIntoDairyFarmDrivewayBlockedByLooseCow with value true or false
- For example, we might notice that a large truck ahead of us is reversing into the driveway of a dairy farm but a cow has got loose and is blocking the truck's path.
- Need to understand the world as having *things* in it that are *related* to each other, not just variables with values.