ARTIFICIAL INTELLIGENCE

LECTURE-2

INTELLIGENT AGENTS

SYED ZUBAIR ALI

INTELLIGENT AGENTS

1 AGENTS AND ENVIRONMENTS

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

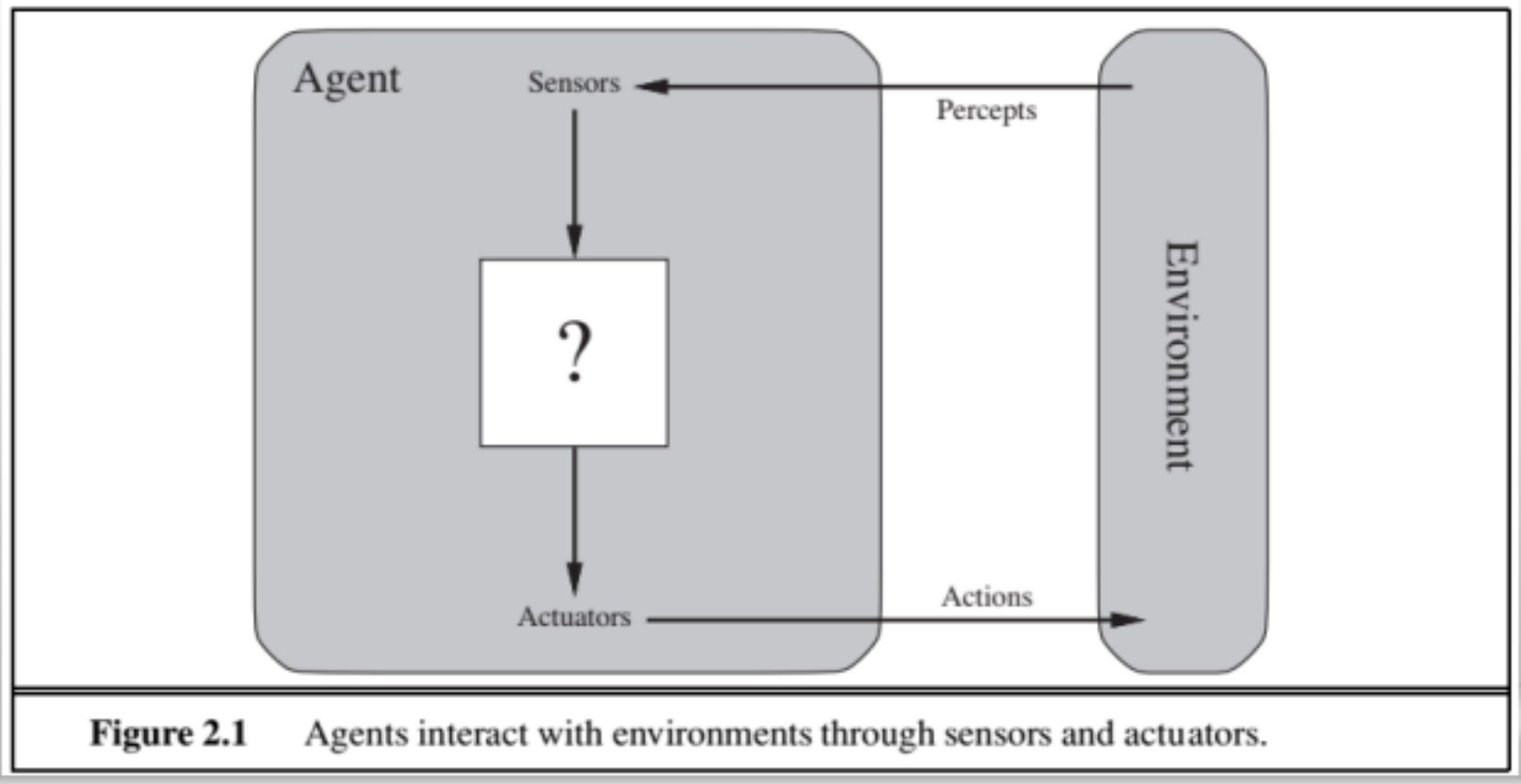


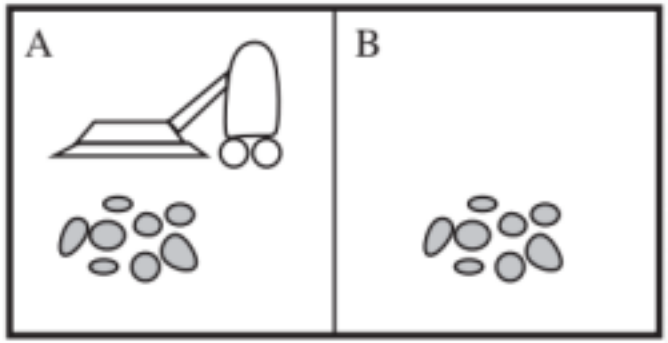
Figure 2.1 Agents interact with environments through sensors and actuators.

INTELLIGENT AGENTS

vacuum-cleaner world

- the vacuum-cleaner world shown in the Figure.
- This world is so simple that we can describe everything that happens.
- This particular world has just two locations: squares A and B.
- The vacuum agent perceives which square it is in and whether there is dirt in the square.
- It can choose to move left, move right, suck up the dirt, or do nothing.
- One very simple agent function is the following: if the current square is dirty, then suck; otherwise, move to the other square.

INTELLIGENT AGENTS



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*
⋮	⋮

INTELLIGENT AGENTS

2 GOOD BEHAVIOR: THE CONCEPT OF RATIONALITY

- A **rational agent** is one that does the right thing—conceptually speaking, every entry in the table for the agent function is filled out correctly.
- Obviously, doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing?

2.1 Rationality

- 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.

This leads to a **definition of a rational agent**:

INTELLIGENT AGENTS

2 GOOD BEHAVIOR: THE CONCEPT OF RATIONALITY

2.2 Omniscience, learning, and autonomy

- Be careful to distinguish between rationality and **omniscience**.
- An **omniscient** agent knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality.

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

INTELLIGENT AGENTS

2 GOOD BEHAVIOR: THE CONCEPT OF RATIONALITY

2.2 Omniscience, learning, and autonomy

- A rational agent not only to gather information but also to **learn** as much as possible 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*.
- To the extent that an agent relies on the prior knowledge of its designer rather than on its own percepts, we say that the agent lacks autonomy.
- A rational agent should be autonomous—it should learn what it can to compensate for partial or incorrect prior knowledge.

INTELLIGENT AGENTS

3 THE NATURE OF ENVIRONMENTS

3.1 Specifying the task environment

- In the rationality of the simple vacuum-cleaner agent, the performance measure, the environment, the agent's actuators and sensors were specified. All these are grouped under the heading of the **task environment**.
- **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.

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.

INTELLIGENT AGENTS

3 THE NATURE OF ENVIRONMENTS

3.2 Properties of task environments

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, then the task environment is **fully observable**.
- A task environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action; relevance, in turn, depends on the performance measure.
- An environment might be **partially observable** because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data—for example, 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**.

INTELLIGENT AGENTS

3 THE NATURE OF ENVIRONMENTS

3.2 Properties of task environments

Single agent vs. **multiagent**:

- An agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two-agent environment.
- In chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's performance measure. Thus, chess is a **competitive** multiagent environment.
- In the taxi-driving environment, on the other hand, avoiding collisions maximizes the performance measure of all agents, so it is a partially **cooperative** multiagent environment.

INTELLIGENT AGENTS

3 THE NATURE OF ENVIRONMENTS

3.2 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.
- If the environment is partially observable, however, then it could appear to be stochastic.
- Most real situations are so complex that it is impossible to keep track of all the unobserved aspects; for practical purposes, they must be treated as stochastic.
- Taxi driving is clearly stochastic in this sense, because one can never predict the behavior of traffic exactly.

INTELLIGENT AGENTS

3 THE NATURE OF ENVIRONMENTS

3.2 Properties of task environments

Episodic vs. **sequential**:

- In an episodic task environment, the agent's experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action. Crucially, the next episode does not depend on the actions taken in previous episodes.
- For example, an agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions; moreover, the current decision doesn't affect whether the next part is defective.
- In sequential environments, the current decision could affect all future decisions.
- 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.

3.2 Properties of task environments

Static vs. dynamic:

- If the environment can change while an agent is deliberating, then we say the environment is **dynamic** for that agent; otherwise, it is **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.
- **Dynamic environments**, on the other hand, are continuously asking the agent what it wants to do; if it hasn't decided yet, that counts as deciding to do nothing. If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is **semidynamic**.
- **Taxi driving is clearly dynamic:** the other cars and the taxi itself keep moving while the driving algorithm dithers about what to do next.
- **Chess**, when played with a clock, is **semidynamic**. Crossword puzzles are **static**.

3 THE NATURE OF ENVIRONMENTS

3.2 Properties of task environments

Discrete vs. continuous:

- The discrete/continuous distinction applies to the *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.
- Taxi-driving actions are also continuous (steering angles, etc.). Input from digital cameras is discrete, strictly speaking, but is typically treated as representing continuously varying intensities and locations.

3 THE NATURE OF ENVIRONMENTS

3.2 Properties of task environments

Known vs. **unknown**:

- In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given.
- Obviously, if the environment is unknown, the agent will have to learn how it works in order to make good decisions.
- The distinction between known and unknown environments is not the same as the one between fully and partially observable environments.

3 THE NATURE OF ENVIRONMENTS

3.2 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.

4 THE STRUCTURE OF AGENTS

agent = architecture + program

4.1 Agent programs

- The agent programs take the current percept as input from the sensors and return an action to the actuators.
- The agent program takes just the current percept as input because nothing more is available from the environment; 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.

4 THE STRUCTURE OF AGENTS

agent = architecture + program

4.1 Agent programs

four basic kinds of agent programs

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

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.

4.2 Simple reflex agents

These agents select actions on the basis of the *current* percept, ignoring the rest of the percept history.

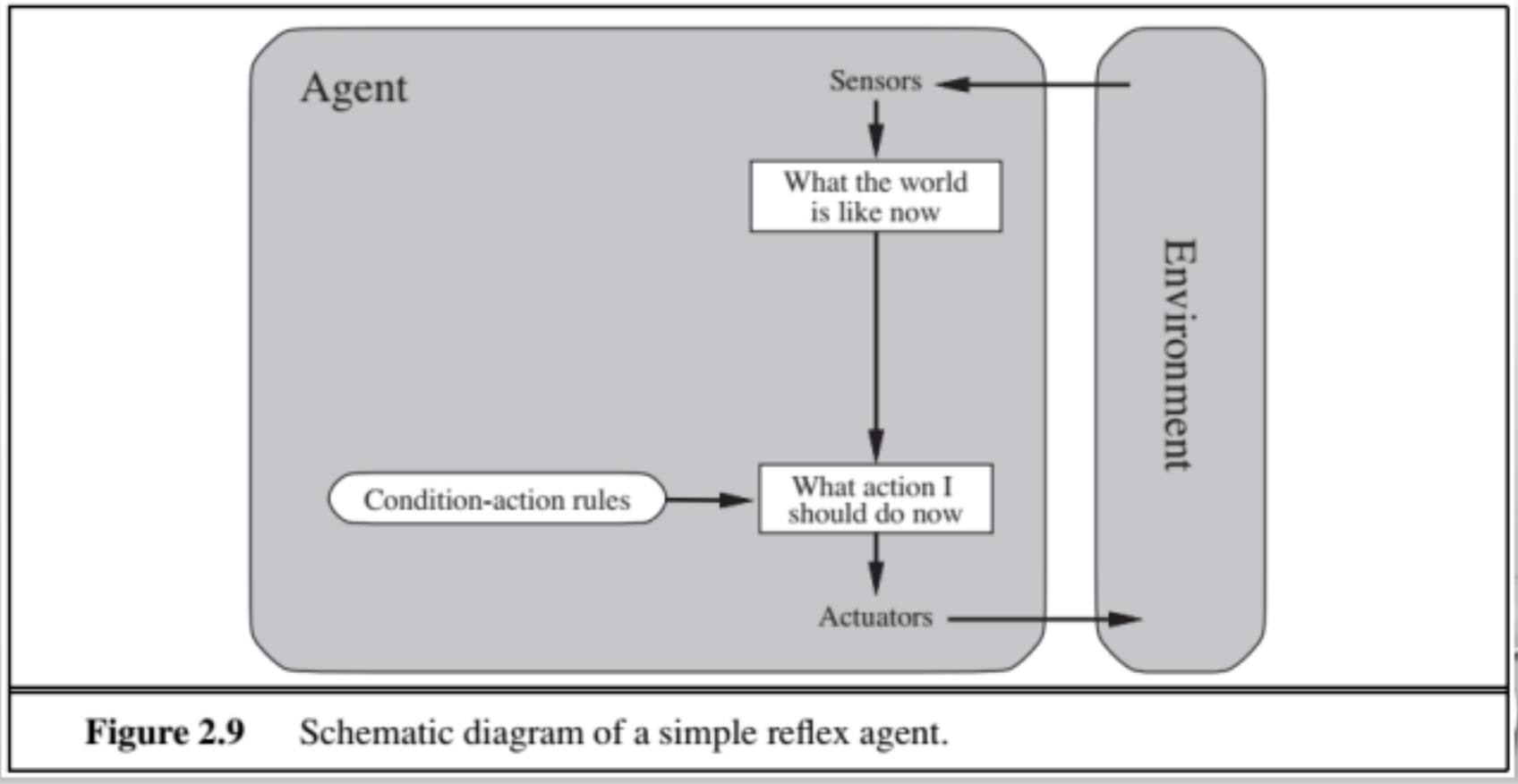


Figure 2.9 Schematic diagram of a simple reflex agent.

4.2 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
```

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

4.3 Model-based reflex agents

- The most effective way to handle partial observability is for the agent to *keep track of the part of the world it can't see now.*
- That is, the agent should 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.
- For the braking problem, the internal state is not too extensive— just the previous frame from the camera, allowing the agent to detect when two red lights at the edge of the vehicle go on or off simultaneously. First, we need some information about how the world evolves independently of the agent—for example, that an overtaking car generally will be closer behind than it was a moment ago.
- Second, we need some information about how the agent's own actions affect the world—for example, that when the agent turns the steering wheel clockwise, the car turns to the right, or that after driving for five minutes northbound on the freeway, one is usually about five miles north of where one was five minutes 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-BASED model is called a model-based agent.

4.3 Model-based reflex agents

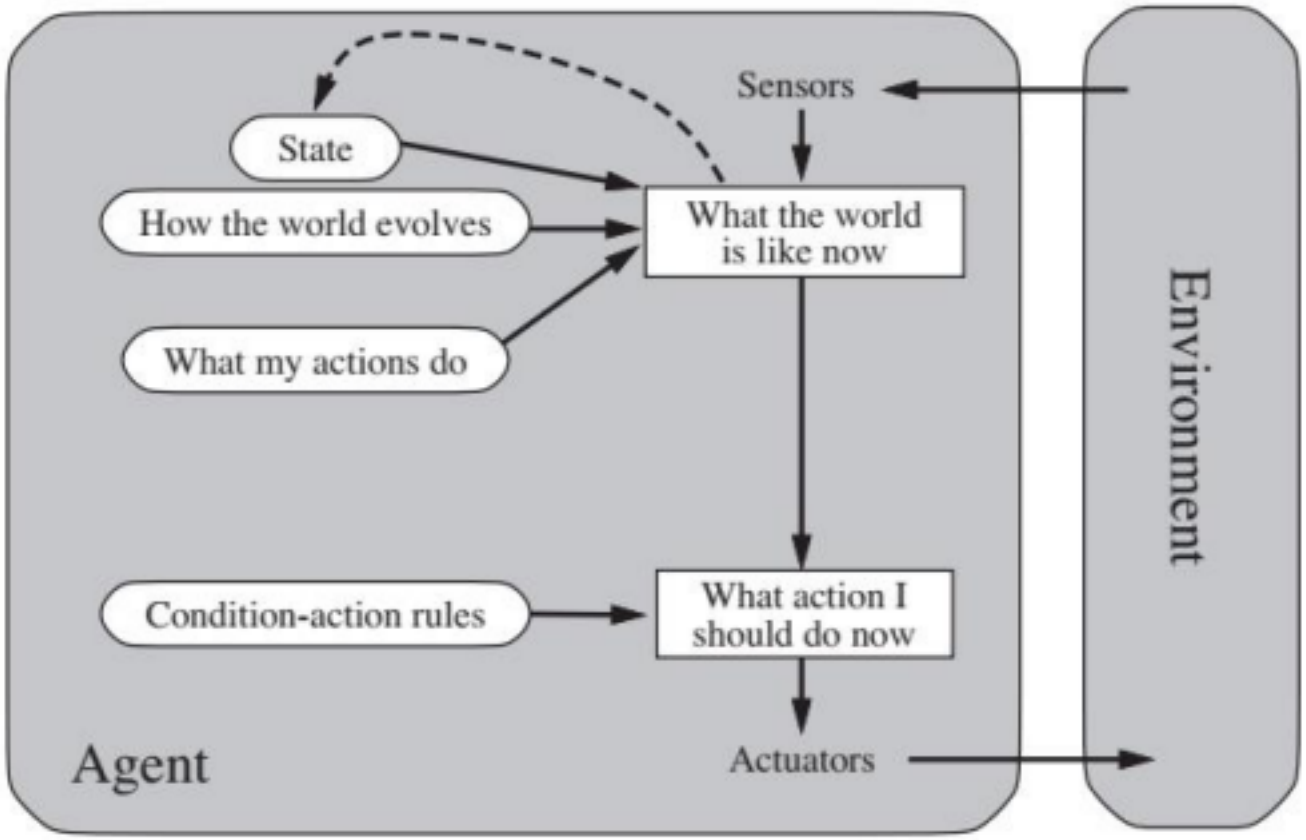


Figure 2.11 A model-based reflex agent.

4.3 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.

4.3 Model-based reflex agents

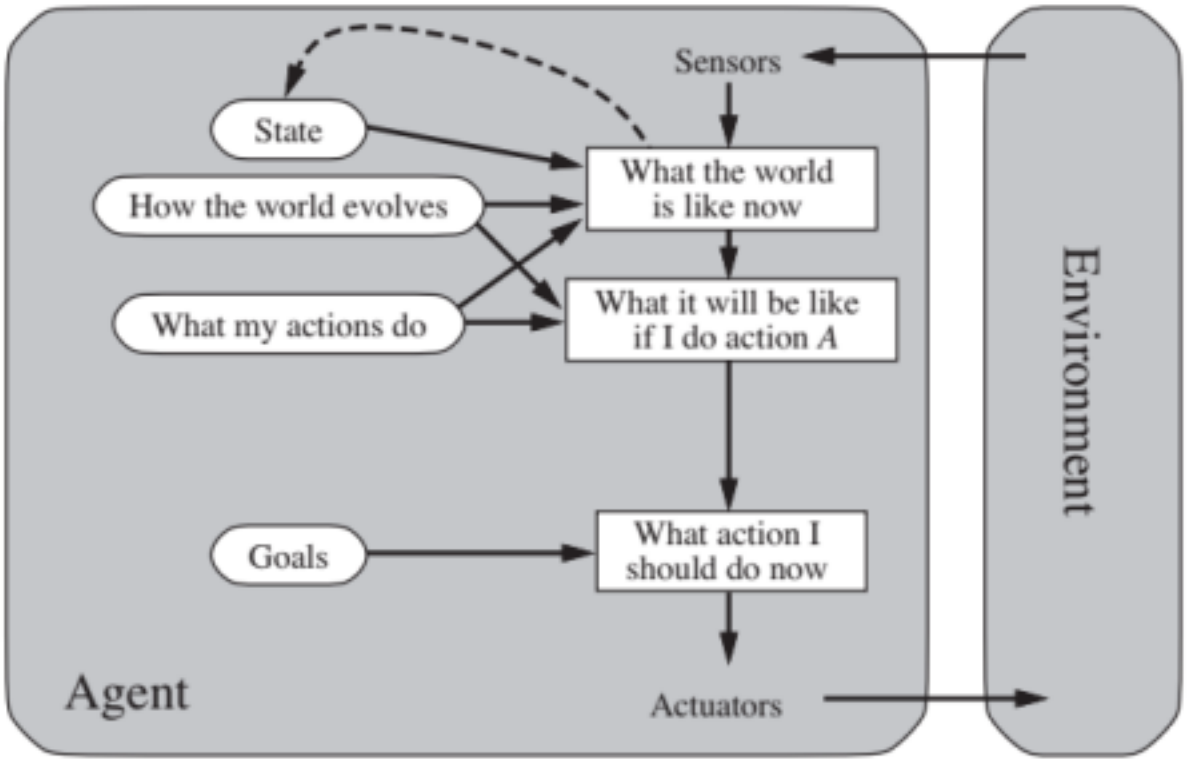


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

4.4 Goal-based agents

- The agent needs some sort of **goal** information that describes situations that are desirable.
- For example, being at the passenger's destination.
- 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.

4.5 Utility-based agents

- Goals alone are not enough to generate high-quality behavior in most environments.
- Goals just provide a crude binary distinction between "happy" and "unhappy" states.
- A more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent. Because "happy" does not sound very scientific, economists and computer scientists use the term utility instead.
- An agent's utility function is essentially an internalization of the 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.

4.5 Utility-based agents

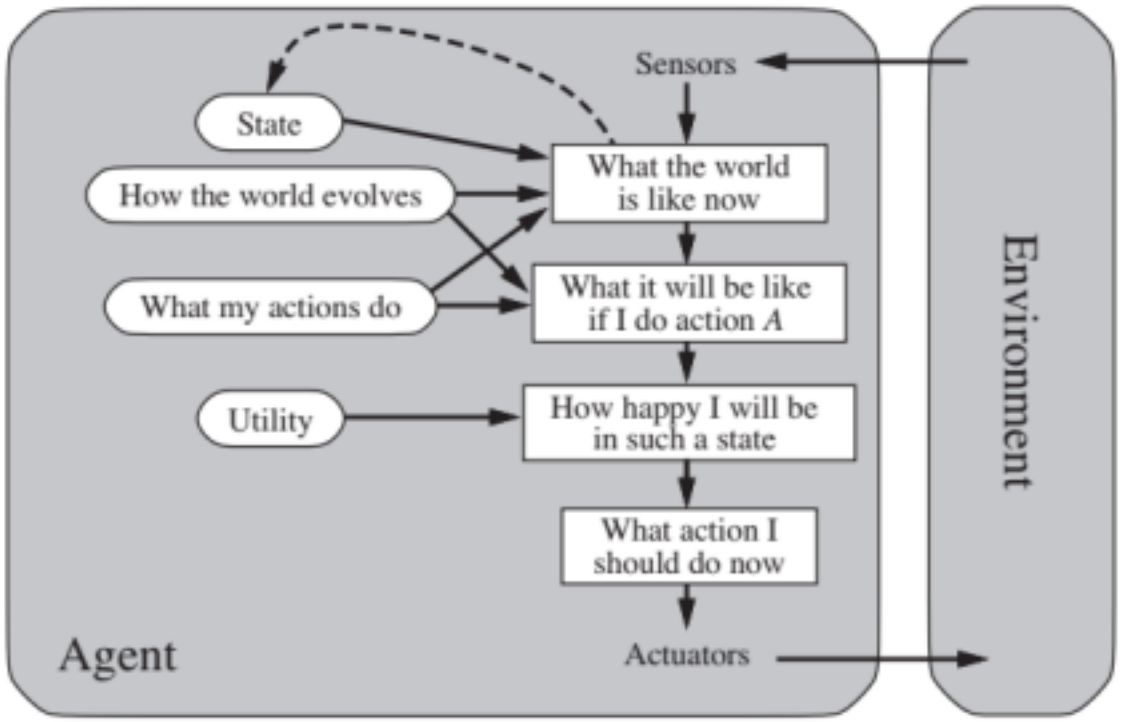


Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome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

4.6 Learning agents

- A learning agent can be divided into four conceptual components, as shown in Figure 2.15.
- The most important distinction is between the **learning element**, which is responsible for making improvements, and the **performance element**, which is responsible for
- selecting external actions.
- The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions.
- The learning element uses feedback from the **critic** on how the agent is doing and determines how the performance element should be modified to do better in the future.

4.6 Learning agents

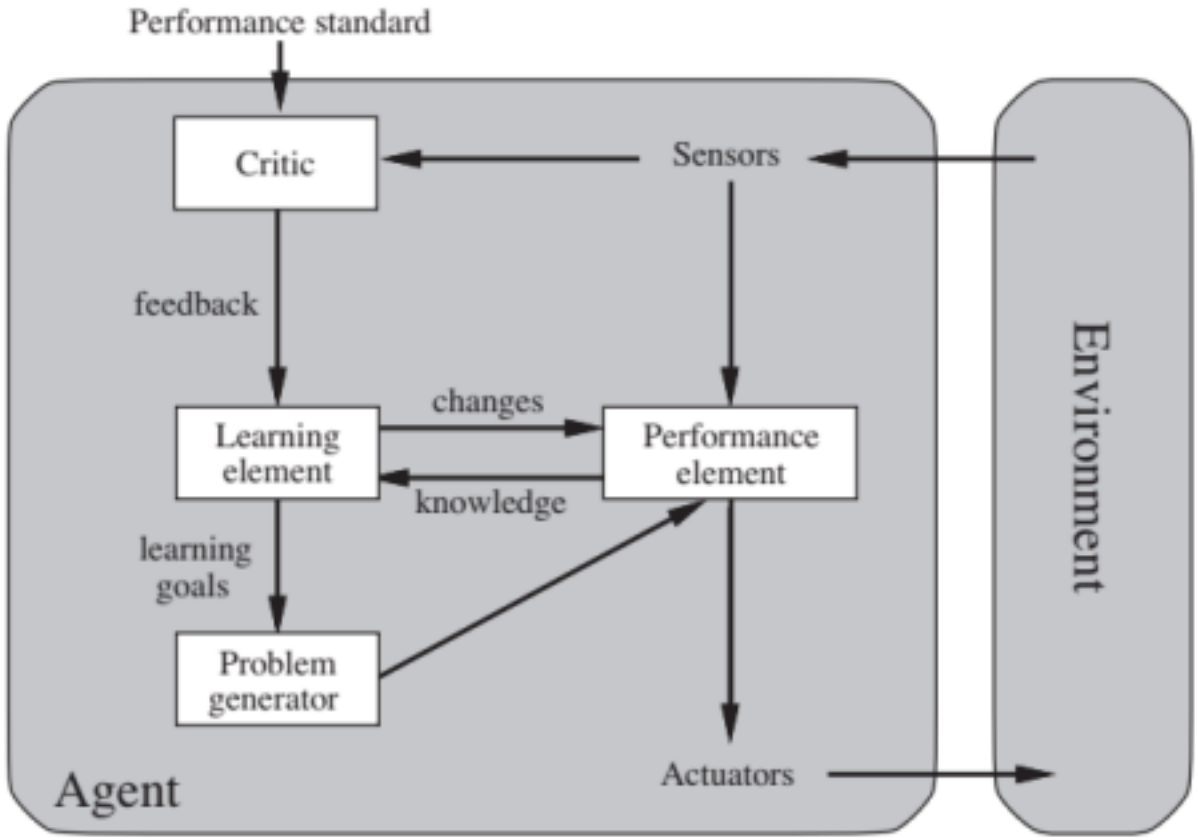


Figure 2.15 A general learning agent.

4.7 How the components of agent programs work

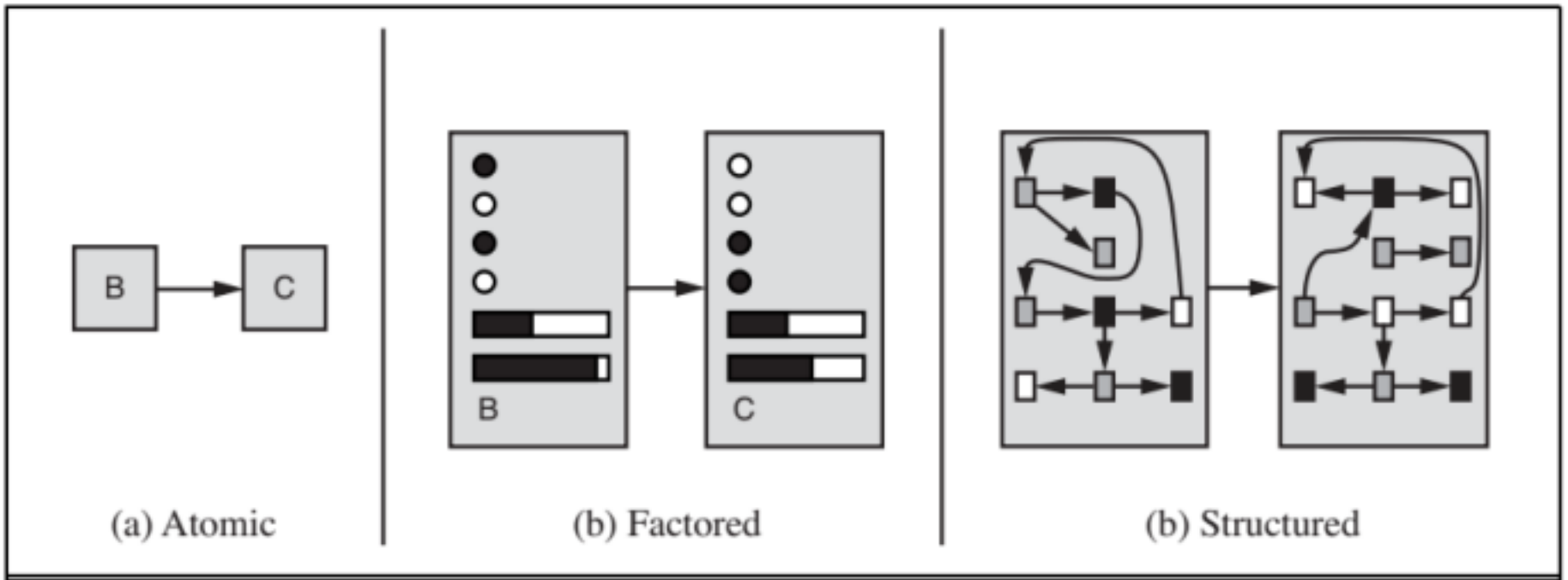


Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed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

4.7 How the components of agent programs work

- In an **atomic representation** each state of the world is indivisible—it has no internal structure.

  A **factored representation** splits up each state into a fixed set of variables or attributes, each of which can have a value.
- The algorithms underlying search and game-playing, Hidden Markov models, and Markov decision processes, all work with **atomic representations.**
- Many important areas of AI are based on **factored representations**, including constraint satisfaction algorithms, propositional logic, planning, Bayesian networks, and the machine learning algorithms.
- A **structured representation**, in which objects such as cows and trucks and their various and varying relationships can be described explicitly.
- **Structured representations** underlie relational databases and first-order logic , first-order robability models, knowledge-based learning and much of natural language understanding.
- In fact, almost everything that humans express in natural language concerns objects and their relationships.

THANK YOU