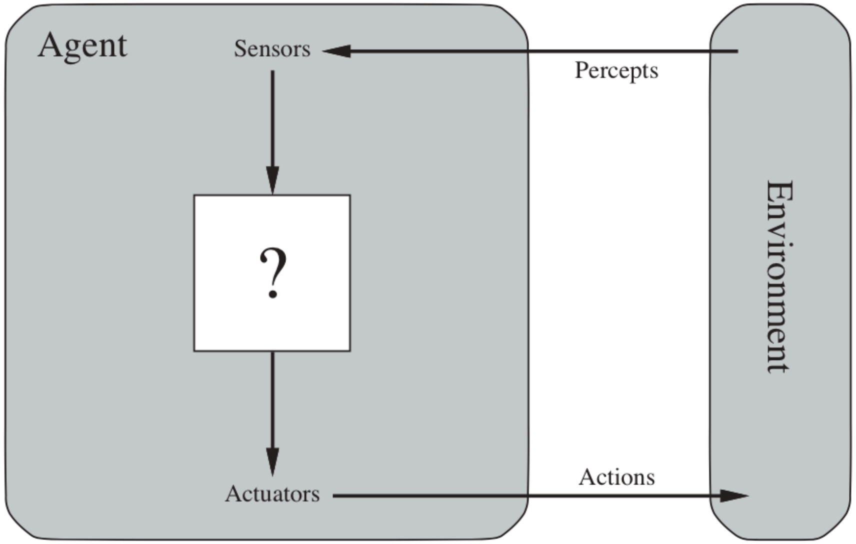
Artificial Intelligence

Dr. Qaiser Abbas
Department of Computer Science & IT,
University of Sargodha
qaiser.abbas@uos.edu.pk

2. Intelligent Agents

- **2.1 AGENTS AND ENVIRONMENTS**
  - An **agent** is anything that can be viewed as **perceiving** its **environment** through **sensors** and acting upon that environment through **actuators**.



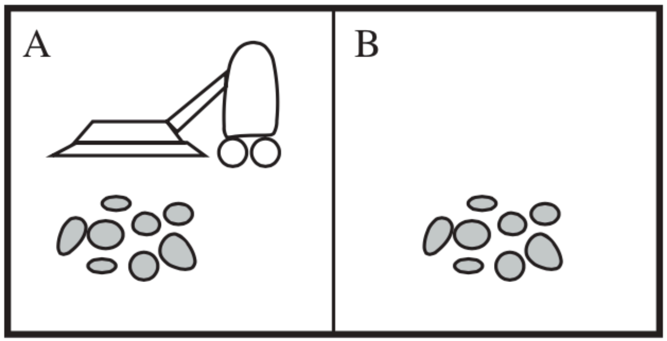
2. Intelligent Agents

- **2.1 AGENTS AND ENVIRONMENTS**
  - The term **percept** refers to the agent's perceptual inputs at any given instant.
  - An agent's **percept sequence** is the complete history of everything the agent has ever perceived.
  - An agent's behavior is described by the **agent function** that maps any given **percept sequence** to an **action**.
  - The **agent program** is a concrete **implementation**, running within some **physical system**.

2.1 AGENTS AND ENVIRONMENTS

- **Example of vacuum-cleaner world**
  – Has just two locations: squares A and B.
  – Perceives which square it is in and whether there is dirt in the square.
  – Choose to move left, right, suck up the dirt, or do nothing.
  – Simple agent function: if the current square is dirty, then suck; otherwise, move to the other square.
  – The pictorial representation is as follows:
    - Vacuum Cleaner World
    - A partial tabulation of this agent function
    - An agent program that implements it

2.1 AGENTS AND ENVIRONMENTS



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

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** $Left$

2.2 GOOD BEHAVIOR: THE CONCEPT OF RATIONALITY

- A **rational agent** is one that does the right thing, but what does it mean to do the right thing?
  - When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives.
  - This sequence of 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.

2.2 GOOD BEHAVIOR: THE CONCEPT OF RATIONALITY

- **Example:** vacuum-cleaner agent from the preceding section.
  - If performance measure: the amount of dirt cleaned up in a single eight-hour shift.
  - A rational agent can maximize this performance measure by cleaning up the dirt, then dumping it all on the floor, then cleaning it up again, and so on.
  - If performance measure: one point could be awarded for each clean square at each time step (perhaps with a penalty for electricity consumed and noise generated).
  - Which one would be better between these two mentioned?
- **General rule: 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.**

2.2.1 Rationality

- What is rational depends on four things:
  - Performance measure : defines the criterion of success.
  - Prior knowledge of the environment.
  - Actions that the agent can perform.
  - 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.

2.2.1 Rationality

- Example: Does the vacuum-cleaner agent function on Slide 5, is a rational agent?
  - That depends on what the **performance measure** is, what is **known about the environment**, and what **sensors and actuators** the agent has. Let us assume the following:
    - If **performance measure** awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
    - If the "geography" of the **environment is known** as *in* Figure of Slide 5 but the dirt distribution and the initial location of the agent are not.
    - Clean squares stay clean and sucking cleans the current square. The Left and Right actions move the agent left and right except when *this would take the agent outside the environment, in which case the agent remains where it is.* Only available actions are **Left, Right, and Suck**.
    - The agent **correctly perceives** its location and whether that location contains dirt.

Under these circumstances the agent is indeed rational;

2.2.2 Omniscience, learning, and autonomy (Read it yourself)

- **Omniscience** is knowing everything, which is not possible.
- **Rationality** maximizes expected performance.
  - For example: **looking both sides** (percept sequence) before crossing the road. Or doing actions in order to modify future percepts—sometimes called **information gathering or exploration** is an important part of rationality e.g. **traffic blocking information and route modification**.

- A rational agent is expected not only to gather information but also to **learn** as much as possible from what it perceives.
  - Prior knowledge of the environment as the agent gains experience may be modified and augmented.
  - In extreme cases, the environment is completely known *a priori*. Then the agent does not need to perceive or learn; it simply acts correctly e.g. lowly dung beetle or female sphex wasp.

2.2.2 Omniscience, learning, and autonomy (Read it yourself)

- If the agent relies on the prior knowledge of its designer rather than on its own percepts, then the agent lacks **autonomy**.

- A rational agent should be autonomous—it should learn what it can, to compensate for partial or incorrect prior knowledge.
  - For example, a vacuum-cleaning agent that learns to foresee where and when additional dirt will appear will do better than one that does not.

2.3 THE NATURE OF ENVIRONMENTS

- **2.3.1 Specifying the task environment**
  - It consists of **PEAS** (**P**erformance, **E**nvironment, **A**ctuators, **S**ensors). All in one group.
  - In designing an agent, the first step must always be to specify the task environment as fully as possible.
  - **Example:** let us consider a more complex problem: an automated taxi driver.

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.

2.3 THE NATURE OF ENVIRONMENTS

– In Figure 2.5, we have sketched the basic PEAS elements for a number of additional agent types.

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.

2.3 THE NATURE OF ENVIRONMENTS

- **Internet shopping agent:**
  - **Performance measure??** Price, quality, appropriateness, efficiency
  - **Environment??** Current and future WWW sites, vendors, shippers
  - **Actuators??** Display to user, follow URL, fill in form
  - **Sensors??** HTML pages (text, graphics, scripts)
- **Question-answering system**
  - **Performance measure??** User satisfaction? Known questions?
  - **Environment??** Wikipedia, Wolfram alpha, ontologies, encyclopedia, . . .
  - **Actuators??** Spoken/written language
  - **Sensors??** Written/spoken input

2.3 THE NATURE OF ENVIRONMENTS

- **2.3.2 Properties of task environments**

  Task environments can be categorized into following.:

  - **Fully observable** vs. **partially observable**:
    - If an agent's sensors give access to the complete state of the environment at each point in time, then we say that the task environment is **fully observable**.
    - 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.
    - If the agent has no sensors at all then the environment is **unobservable**.

2.3 THE NATURE OF 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, avoiding collisions maximizes the performance measure of all agents, so it is a partially **cooperative** multiagent environment.
- **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, then it could appear to be stochastic but next state may not be predicted precisely.

2.3 THE NATURE OF ENVIRONMENTS

- **Example:** Taxi driving is clearly stochastic in this sense, because one can never predict the behavior of traffic exactly; moreover, one's tires blow out and one's engine seizes up without warning.
- An environment is **uncertain** if it is not fully observable or not deterministic.
- "**Stochastic**" generally implies that uncertainty about outcomes is quantified in terms of probabilities;
- A **nondeterministic** environment is one in which actions are characterized by their possible outcomes, but no probabilities are attached to them.

– **Episodic** vs. **sequential**:

- In an **episodic task environment**, the agent's experience is divided into episodes. In each episode the agent receives a percept and then performs a single action. Crucially (critically or decisively), 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**, on the other hand, 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.

2.3 THE NATURE OF ENVIRONMENTS

- **Static** vs. **dynamic**:
  - If the environment can change while an agent is deliberating (engage in long and careful consideration), then we say the environment is dynamic for that agent; otherwise, it is static. (Read it Yourself)
- **Discrete** vs. **continuous**: (Read it yourself)
  - 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.
  - A discrete environment has fixed locations or time intervals. A continuous environment could be measured quantitatively to any level of precision.
- **Known** vs. **unknown**: (Read it yourself)
  - This distinction refers not to the environment itself but to the agent's state of knowledge.
  - In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given.
  - If the environment is unknown, the agent will have to learn how it works in order to make good decisions.

2.3 THE NATURE OF ENVIRONMENTS

- **Environment class:** to evaluate a taxi driver in simulated traffic, we would want to run many simulations with different traffic, lighting, and weather conditions called environment class.

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.

2.4 THE STRUCTURE OF AGENTS

agent = architecture + program

- Program runs on some sort of computing device with physical sensors and actuators—we call this the **architecture**

- The job of AI is to design an **agent program** that implements the agent function— the mapping from percepts to actions.

2.4 THE STRUCTURE OF AGENTS

- **2.4.1 Agent programs**
  - To build a rational agent, we as designers must construct a table that contains the appropriate action for every possible percept sequence.
  - Figure 2.7 shows a rather trivial (little importance) agent program that keeps track of the percept sequence and then uses it to index into a table of actions to decide what to do.

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.

2.4 THE STRUCTURE OF AGENTS

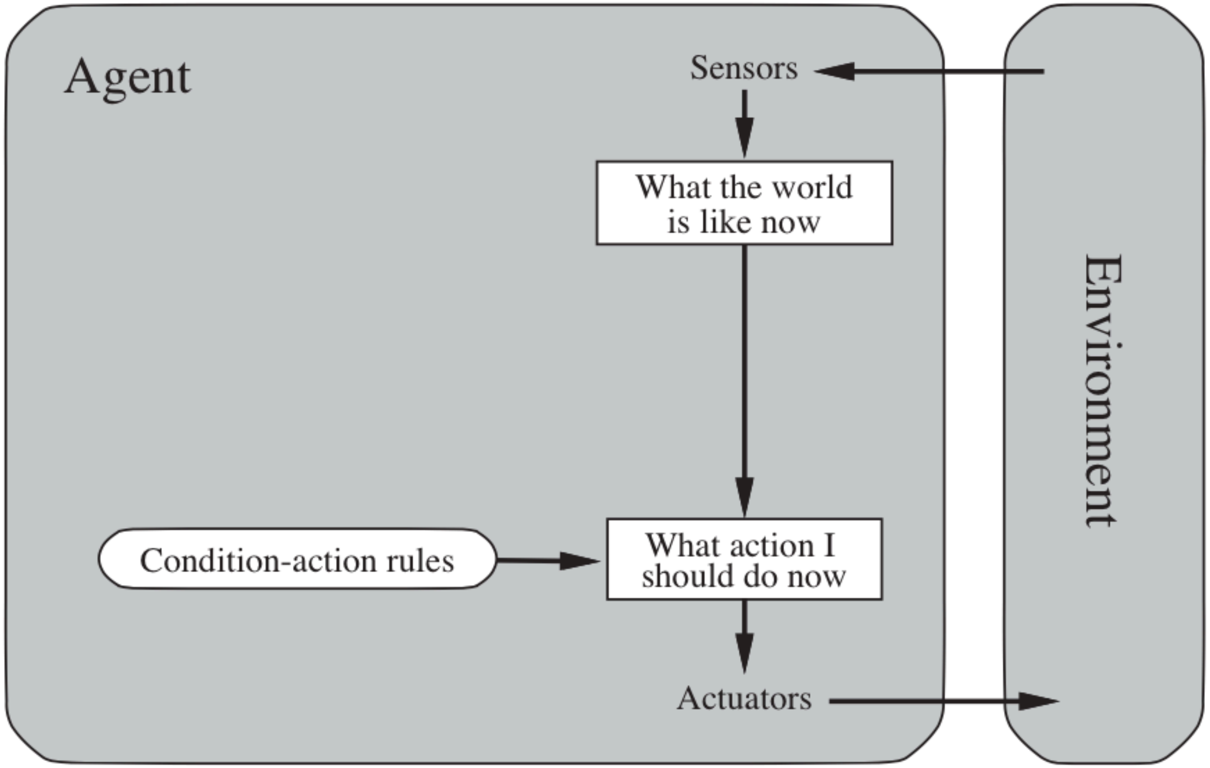
- **Drawbacks of table-driven approach:**
  - Table-driven approach to agent construction is doomed (unfortunate) to failure.
  - Let P be the set of possible percepts and let T be the lifetime of the agent (the total number of percepts it will receive). The lookup table will contain $\sum_{t=1}^{T} |P|^t$ entries.
  - Daunting (difficult to deal) size of these tables.
  - No physical agent in this universe will have the space to store the table.
  - The designer would not have time to create the table,
  - No agent could ever learn all the right table entries from its experience means no autonomy.
  - Even with learning, need a long time to learn the table entries.

2.4 THE STRUCTURE OF AGENTS

- The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behavior from a smallish program rather than from a vast table.
- Four basic kinds of agent programs are as follows:
  - Simple reflex agents;
  - Model-based reflex agents;
  - Goal-based agents; and
  - Utility-based agents.

2.4 THE STRUCTURE OF AGENTS

- **2.4.2 Simple reflex agents**
  - These agents select actions on the basis of the *current* percept, ignoring the rest of the percept history. For example, the vacuum agent whose agent function is tabulated in Figure on slide 5 is a simple reflex agent, because its decision is based only on the current location and on whether that location contains dirt. An agent program for this agent is shown in Figure on slide 5.

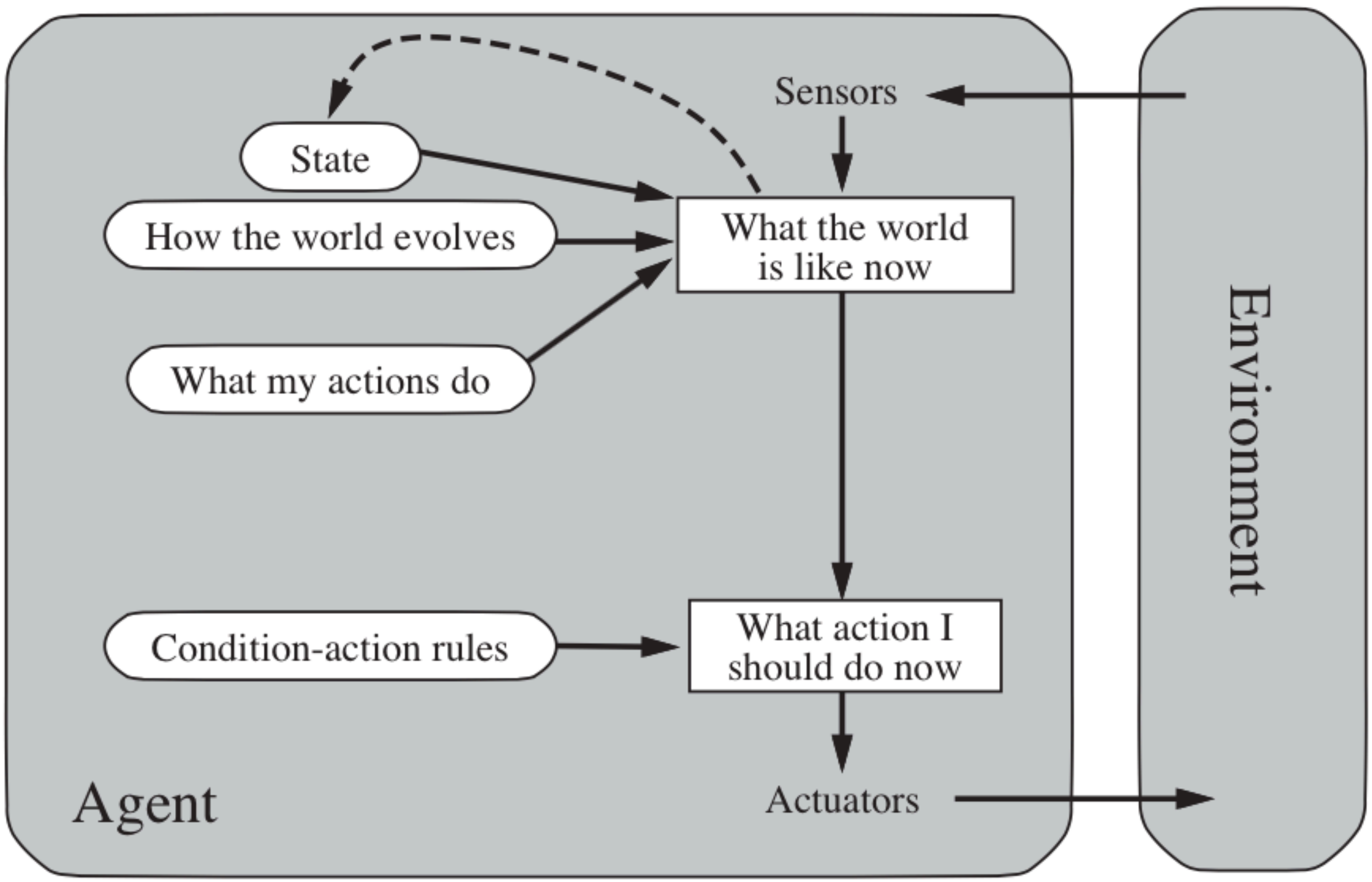


2.4 THE STRUCTURE OF AGENTS

- **2.4.3 Model-based reflex agents**
  - In this, 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.
    - **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.
    - **How the world evolves?:** For example, an overtaking car generally will be closer behind than it was a moment ago.
    - **What my actions do?:** After driving for five minutes towards Makkah on the Highway, one is usually about five miles towards Makkah of where one was five minutes ago.

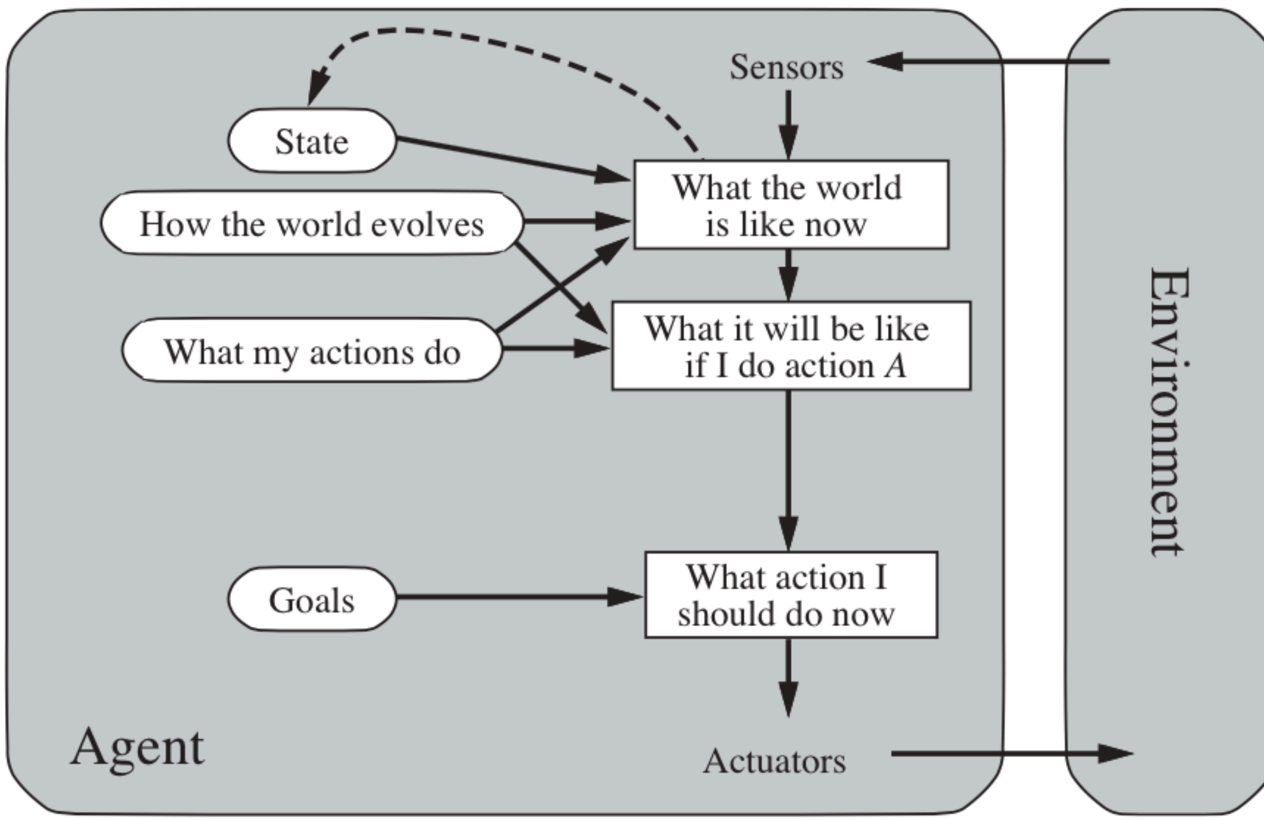
2.4 THE STRUCTURE OF AGENTS

- **2.4.3 Model-based reflex agents**



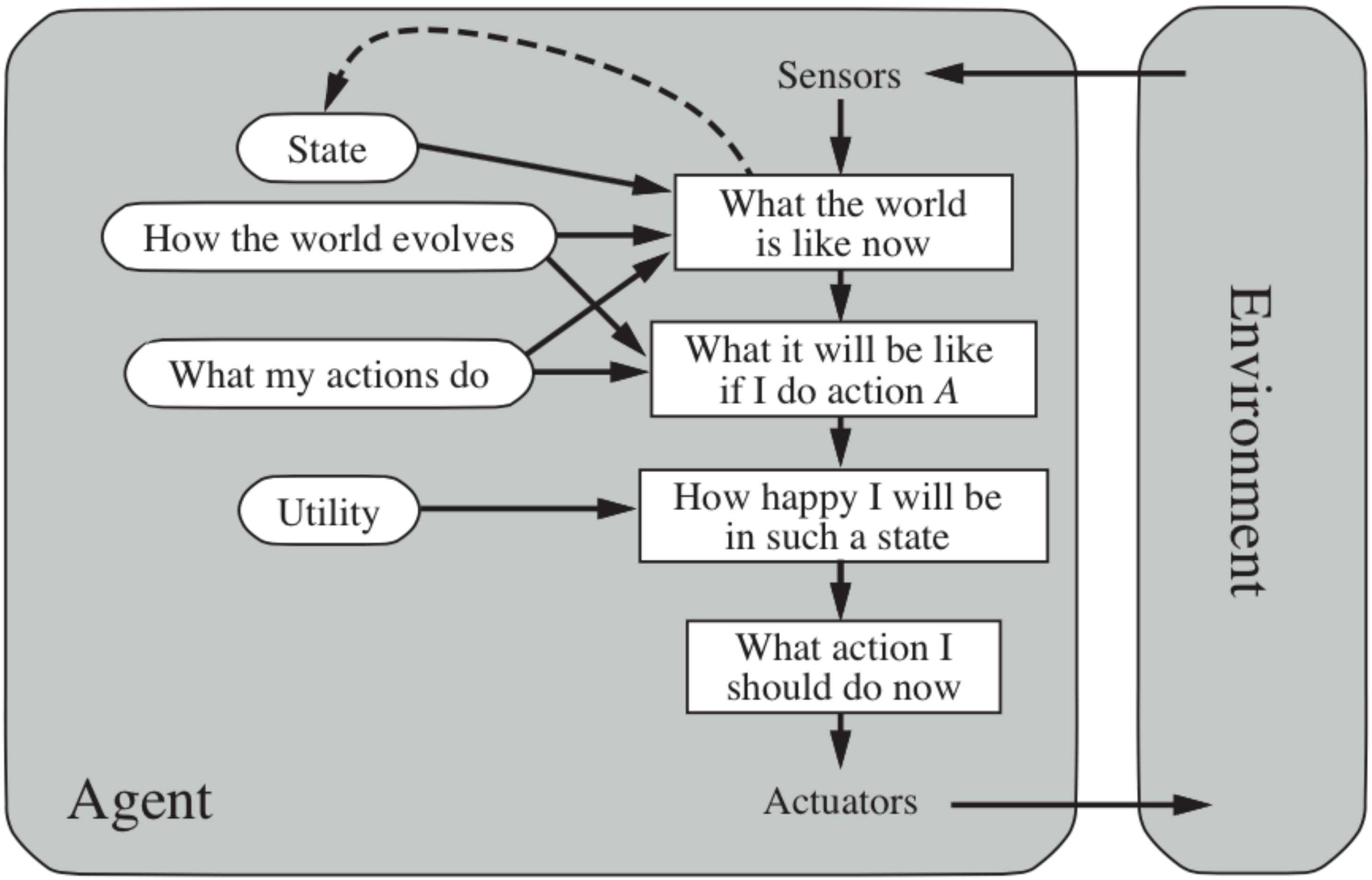
2.4 THE STRUCTURE OF AGENTS

- **2.4.4 Goal-based agents**
  - The agent needs some sort of **goal** information that describes situations that are desirable—For example, at a road junction, the taxi can turn left, turn right, or go straight on. The correct decision depends on where the taxi is trying to get to.
  - The agent program can combine this with the model- based reflex agent to choose actions that achieve the goal.



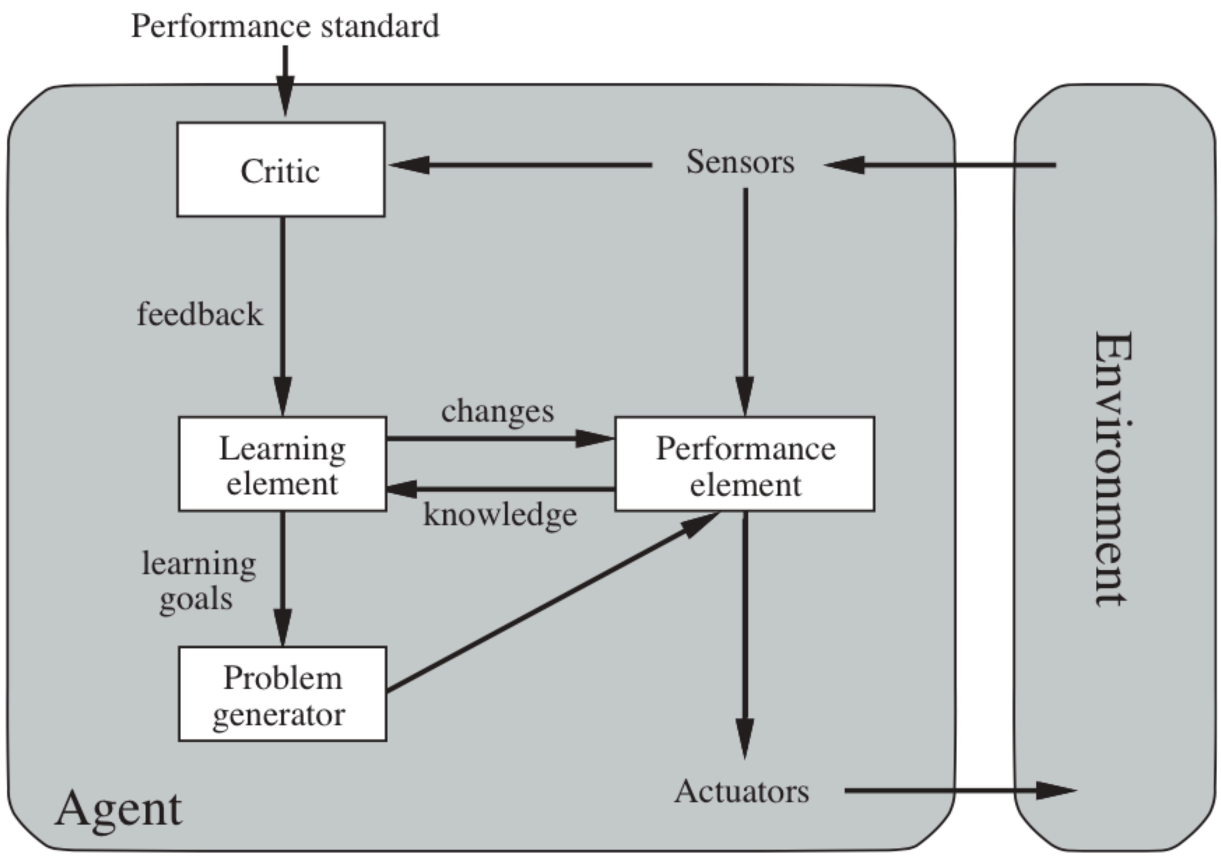
2.4 THE STRUCTURE OF AGENTS

- **2.4.5 Utility-based agents**
  - Utility-based agents addresses the partial observability and stochasticity and decision making under uncertainty.



2.4 THE STRUCTURE OF AGENTS

- **2.4.6 Learning agents**
  - Learning allows the agent to operate in initially unknown environments.
  - **Performance element** is responsible for selecting external actions.
  - **Learning element** is responsible for making improvements. It takes feedback from the **critic** on how the agent is doing and determines changes for the performance element to do better in the future.
  - **Critic** tells the learning element how well the agent is doing with respect to a fixed performance standard.
  - If the agent is willing to explore a little and do some suboptimal actions in the short run, to help much better actions for the long run. The **problem generator**'s job is to suggest these exploratory actions.



Assignment No.2

1. Understand the Vacuum Cleaner World given at the following link:

   http://web.ntnu.edu.tw/~tcchiang/ai/Vacuum%20Cleaner%20World.htm
2. Download the source code, configure it and execute it. After having a hands on experience, submit the report with complete discussion of steps you made to make this source code executable along with the explanation of handmade output screenshots.
3. Be ready for your presentation at any time.