

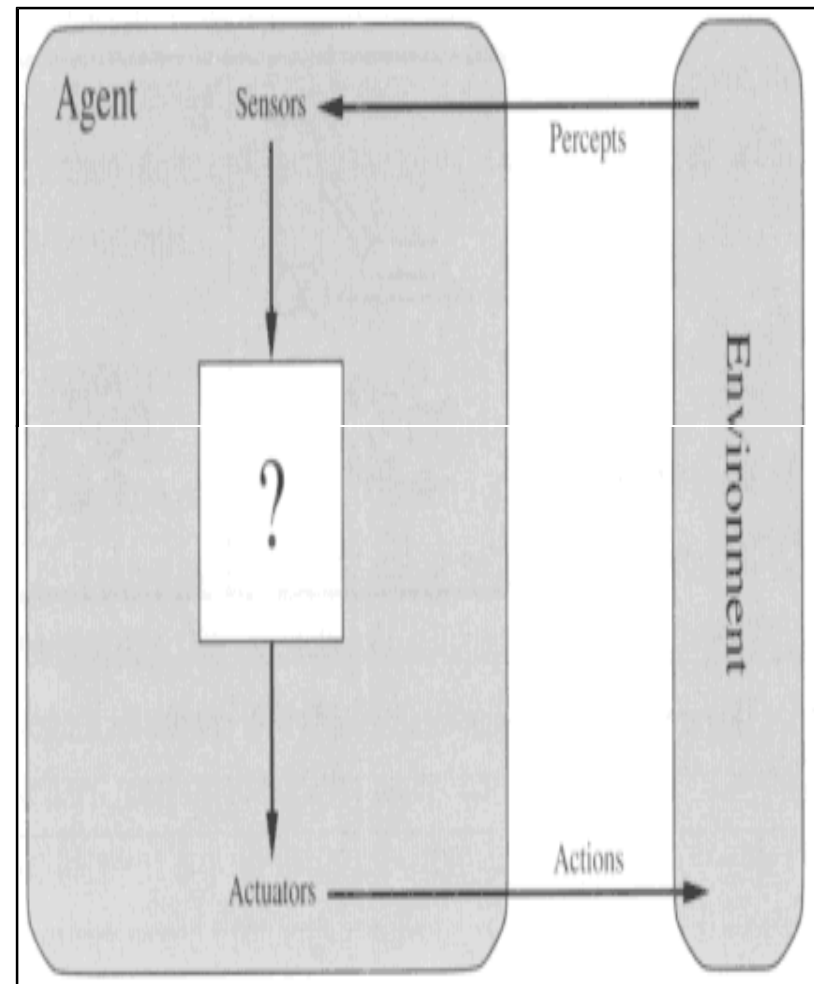
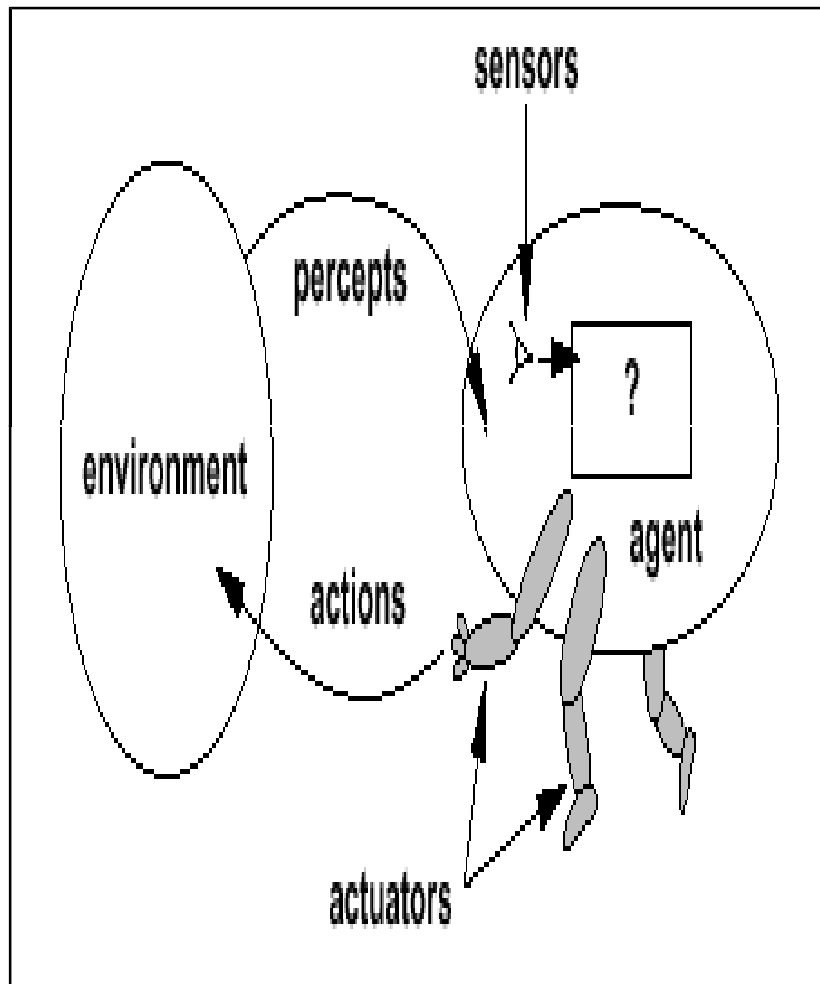
INTELLIGENT AGENTS



AGENTS

- ✧ An agent is anything that can perceive its **environment** through **sensors** and acting upon its environment through **actuators**.
- ✧ **Human agent** has eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for actuators
- ✧ **Robotic agent** has cameras and infrared range finders for sensors; various motors for actuator.
- ✧ **Software agent** receives keystrokes, file contents and network packets as sensors and acts on the environment by displaying on screen, writing files, and sending network packets.
- ✧ Every agent can perceives its own actions (but **not** always the effects).

AGENTS CONTD....



PERCEPT & PERCEPT SEQUENCE

▢ Percept

▢ Agent's perceptual inputs at any given instant

▢ Percept Sequence

▢ Complete history of everything that the agent has ever perceived.

▢ An agent choice of action at any given instant can depend on the entire percept sequence observed to date.

▢ Thus, to describe an agent it is necessary to specify the **agent's choice of action** for every **possible percept sequence**.

AGENT FUNCTION & AGENT PROGRAM

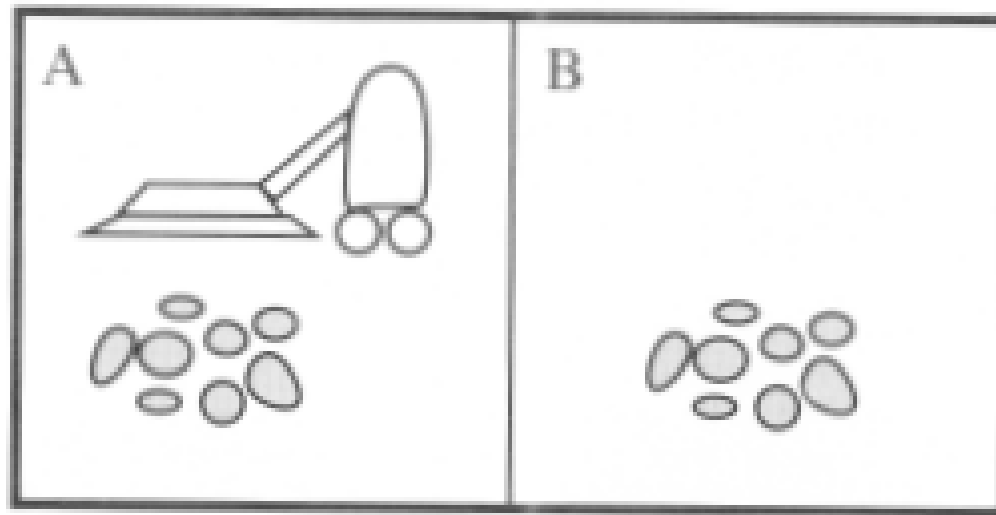
- ✧ The agent behavior can be mathematically described by agent function.
- ✧ **An agent function** maps any given percept sequence to an action.

$$f: P^* \rightarrow A$$

- ✧ One way of describing agent function is to tabulate all possible sequences and recording which actions the agent does in response.
- ✧ But in real time situations, the entries in the table are infinite and not possible to tabulate all of them.
- ✧ The agent function for the agent is internally implemented with the **agent program**.
- ✧ The agent function is an abstract mathematical description whereas the agent program is concrete representation, running on the agent architecture.

THE VACUUM-CLEANER WORLD-EXAMPLE

- ✧ Consider a vacuum cleaner task. It's also a made up-world and many variations can be made in it.
- ✧ This particular world has just two locations: square A and square B.
- ✧ Perception: Clean or Dirty? where it is in?
- ✧ Actions: Move left, Move right, suck, do nothing



THE VACUUM-CLEANER WORLD-EXAMPLE

CONTD....

- One very simple agent function is that if the current square is dirty then suck the dirt else move to the other square.

Precept 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



AGENT PROGRAM- VACUUM CLEANER WORLD

Function Reflex-Vacuum-Agent(*[location,status]*)

return an action

If *status* = *Dirty* **then return** *Suck*

else if *location* = *A* **then return** *Right*

else if *location* = *B* **then return** *left*



CONCEPT OF RATIONALITY

▢ Rational Agent

- ▢ One that does the right thing
- ▢ every entry in the table for the agent function is correct (rational).

▢ What is correct?

- ▢ The actions that cause the agent to be most successful
- ▢ So we need ways to measure success.



PERFORMANCE MEASURE

- ⌞ An agent, based on its percepts perform sequence of actions. If the actions are desirable, it is said to be performing well.
- ⌞ Performance measure
 - An objective function that determines
 - ⌞ How the agent does successfully
 - ⌞ E.g., 90% or 30%?
- ⌞ No universal performance measure for all agents.



PERFORMANCE MEASURE

✧ A general rule to design performance measures according to

- What one actually wants in the environment
- Rather than how one thinks the agent should behave

✧ E.g., in vacuum-cleaner world

- We want the floor clean, no matter how the agent behaves
- We don't restrict how the agent behaves
- performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.



RATIONALITY

What is rational at any given time depends on four things:

- The performance measure defining the criterion of success.
- The agent's prior knowledge of the environment.
- The actions that the agent can perform.
- The agent's percept sequence up to now.
- **For each possible percept sequence,**
 - an rational agent should select
 - an action expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has

SPECIFYING TASK ENVIRONMENT

- ✧ In designing an agent, the first step must always be to specify the task environment as fully as possible.
- ✧ The task environment is specified using four parameters.
 - ✧ Performance
 - ✧ Environment
 - ✧ Actuators
 - ✧ Sensors
- ✧ These four parameters are called as PEAS (**P**erformance, **E**nvironment, **A**ctuator, **S**ensor) specification.



PEAS-AUTOMATED TAXI DRIVER

▢ Performance measure

▢ How can we judge the automated driver?

▢ Which factors are considered?

- ▢ getting to the correct destination
- ▢ minimizing fuel consumption
- ▢ minimizing the trip time and/or cost
- ▢ minimizing the violations of traffic laws
- ▢ maximizing the safety and comfort, etc.

▢ Environment

- A taxi must deal with a variety of roads.
- Traffic lights, other vehicles, pedestrians, stray animals, road works, police cars, etc.
- Interact with the customer



PEAS-AUTOMATED TAXI DRIVER CONTD...

▣ Actuators (for outputs)

- Control over the accelerator, steering, gear shifting and braking
- A display to communicate with the customers

▣ Sensors (for inputs)

- Detect other vehicles, road situations
- GPS (Global Positioning System) to know where the taxi is
- Many more devices are necessary such as speedometer, engine sensors, keyboards, sonar,



PEAS-MEDICAL DIAGNOSIS SYSTEM

- ⌘ Performance measure: Healthy patient, minimize costs, lawsuits
- ⌘ Environment: Patient, hospital, staff
- ⌘ Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
- ⌘ Sensors: Keyboard (entry of symptoms, findings, patient's answers)



PEAS-PART PICKING ROBOT

- ▮ Performance measure: Percentage of parts in correct bins
- ▮ Environment: Conveyor belt with parts, bins
- ▮ Actuators: Jointed arm and hand
- ▮ Sensors: Camera, joint angle sensors



PEAS- ONLINE ENGLISH TUTOR

- ⌘ Performance measure: Maximize student's score on test
- ⌘ Environment: Set of students
- ⌘ Actuators: Screen display (exercises, suggestions, corrections)
- ⌘ Sensors: Keyboard



TASK ENVIRONMENT

✧ An agent operates in a **task environment**: –

Task: the goal(s) the agent is trying to achieve.

Environment: that part of the real world or a computational system 'inhabited' by the agent.



Types OF ENVIRONMENTS

✚ An agent may have to act on the following types of tasks environments.

- Fully observable vs. Partially observable
- Deterministic vs. Stochastic
- Episodic vs. Sequential
- Static vs. Dynamic
- Discrete vs. Continuous
- Single agent vs. multiagent



Fully Observable vs. Partially Observable



Fully Observable



Partially Observable



Fully Observable vs. Partially Observable

- ✧ The task environment is fully observable if an agent's sensors give it access to the complete state of the environment.
- ✧ For fully observable environment, the sensors should detect all the aspects that are relevant to the choice of action (relevance depends upon the performance measure).
- ✧ Fully observable environments are desirable and convenient as the agent need not to maintain internal state to keep the track of the world.
- ✧ The environment of crossword puzzle, chess, image analysis are fully observable.



Fully Observable vs. Partially Observable Contd...

✚ An environment is partially observable if the entire state of the environment is not known to the agent due to inaccurate sensors, missing states from sensor data, noise, etc.

✚ For instance,

- a vacuum cleaner *Partially Observable*



- In self driving car, *Partially Observable*

An agent knows about every sensor and actuator present in the car but have no idea about the road condition, no. of vehicles on the road, etc.

DETERMINISTIC VS. STOCHASTIC

- ✧ An environment is deterministic, if the next state of the environment is completely determined by the **current state** and **action** executed by the agent.
- ✧ An agent need not to deal with uncertainty in a fully observable, deterministic environment.
- ✧ In chess , the agents know the aftereffects of any action.



DETERMINISTIC VS. STOCHASTIC CONTD....

- ✧ An environment is stochastic, if the next state of the environment is not completely determined by the current state and action executed by the agent.
- ✧ Taxi driving is clearly stochastic
one can never predict the behavior of the traffic exactly, some faults arise in the taxi without warning, etc.
- ✧ Interactive English Tutor are also examples of stochastic environments.
- ✧ If the environment is deterministic except for the action of the other agents, then the environment is **strategic** such as a chess or a poker playing agent.



EPISODIC VS. SEQUENTIAL

- In an episodic task environment, the agent's experience is divided into atomic episodes.
- The choice of action depends only on the episode itself.



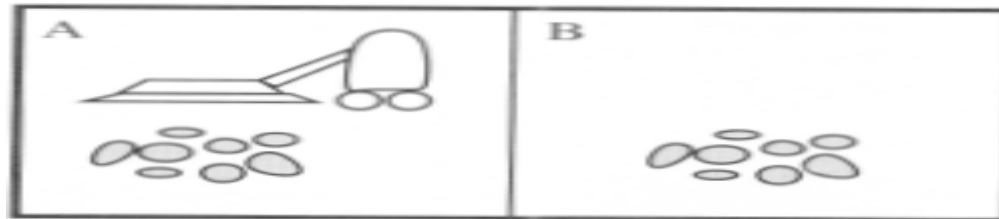
- Many classification tasks are episodic.

For instance,

- Find defective bottles,



- Vaccum cleaner agent is episodic in nature,



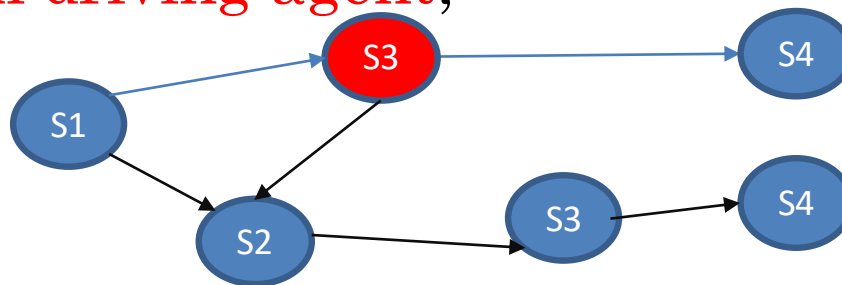
EPISODIC VS. SEQUENTIAL CONTD...

✧ In sequential environments, the current decision could affect all future decisions.

✧ In chess,

The choice of one action may dictate the next action to be chosen

✧ For a taxi driving agent,



✧ In both cases, short term actions have long-term consequences.

✧ Episodic environments are much simpler than sequential agents because the agent does not need to think ahead.

STATIC VS. DYNAMIC

- ✧ If the environment can change while an agent is deliberating , then the environment is dynamic; otherwise it is static.
- ✧ For Instance,
 - *Taxi driving* is clearly dynamic as the other cars are also moving while the driving algorithm decides what to do next.
 - *Cross word puzzle* is an example of static environment.
- ✧ Static environments are easy to deal with because the agent need not to keep looking at the world.



STATIC VS. DYNAMIC CONTD...

- ✚ If the environment does not change with the passage of time but agent's performance score does then the environment is **semi-dynamic**.
- ✚ Chess is the example of semi-dynamic environments.

The performance of the Agent may be degraded if the agent takes more time to complete the action.



DISCRETE VS. CONTINUOUS

- ✧ An environment is said to be discrete if there are a limited number of distinct states, clearly defined percepts and actions.
- ✧ A chess game has discrete environment as it has finite number of distinct states and a discrete set of percepts and actions.
- ✧ In continuous environment, the states, percepts or actions are continuous time problems.
- ✧ For instance, in case of taxi driving, the **speed** and **location of taxi** and other vehicles are continuously changing.
- ✧ The taxi driving actions are also continuous such as (steering, angles, etc.)
- ✧ Input from cameras is discrete, but it is typically treated as continuous with varying intensities and locations.

SINGLE AGENT VS. MULTI AGENT

- ✧ If only one agent is involved in the task then the environment is single agent other wise it is multi agent.
- ✧ For instance,
Crossword puzzle is a single agent scenario
Chess or Taxi driving is a multi agent scenario.



SINGLE AGENT VS. MULTI AGENT

- ✧ A multi agent environment can be **competitive** or **cooperative**.
- ✧ *In a competitive environment*, the opponent agent tries to maximize its performance measure thereby minimizing the agent's performance score.

Chess is a competitive multi agent environment.

- ✧ *In a cooperative environment*, the agent tries to maximize the performance measure of all the agents in the environment.

Taxi driving is an example of co- operative multi agent environment.



STRUCTURE OF AGENTS

✧ Agent = architecture + program

- Architecture = some sort of computing device (sensors + actuators)
- (Agent) Program = some function that implements the agent mapping.

✧ In general, architecture makes the percept available from the sensors to the program, runs the program and feeds the program action choice to the actuators as they are generated.

✧ Agent Program = Job of AI



AGENT PROGRAMS

- ✧ Input for Agent Program
 - Only the current percept
- ✧ Input for Agent Function
 - The entire percept sequence
 - The agent must remember all of them
- ✧ One way to implement the agent program is to maintain a table look up (agent function).



TABLE-DRIVEN AGENT PROGRAM

✎ **Function:** TABLE-DRIVEN-AGENT (percept)
returns an action

static: *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 \leftarrow *Lookup(percepts, table)*
return **action**



TABLE-DRIVEN AGENT PROGRAM CONTD....

- ▢ P = the set of possible percepts
- ▢ T = lifetime of the agent
 - ▢ The total number of percepts it receives
- ▢ Size of the look up table

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

- ▢ Consider playing chess
 - ▢ P = 10, T = 150
 - ▢ Will require a table of at least 10^{150} entries



TABLE-DRIVEN AGENT PROGRAM CONTD....

- ▮ Despite of huge size, look up table does what we want.
- ▮ The key challenge of AI
 - ▮ Find out how to write programs that, to the extent possible, produce rational behavior
 - ▮ Rather than a large amount of table entries
- ▮ E.g., a five-line program of Newton's Method vs. huge tables of square roots, sine, cosine, ...



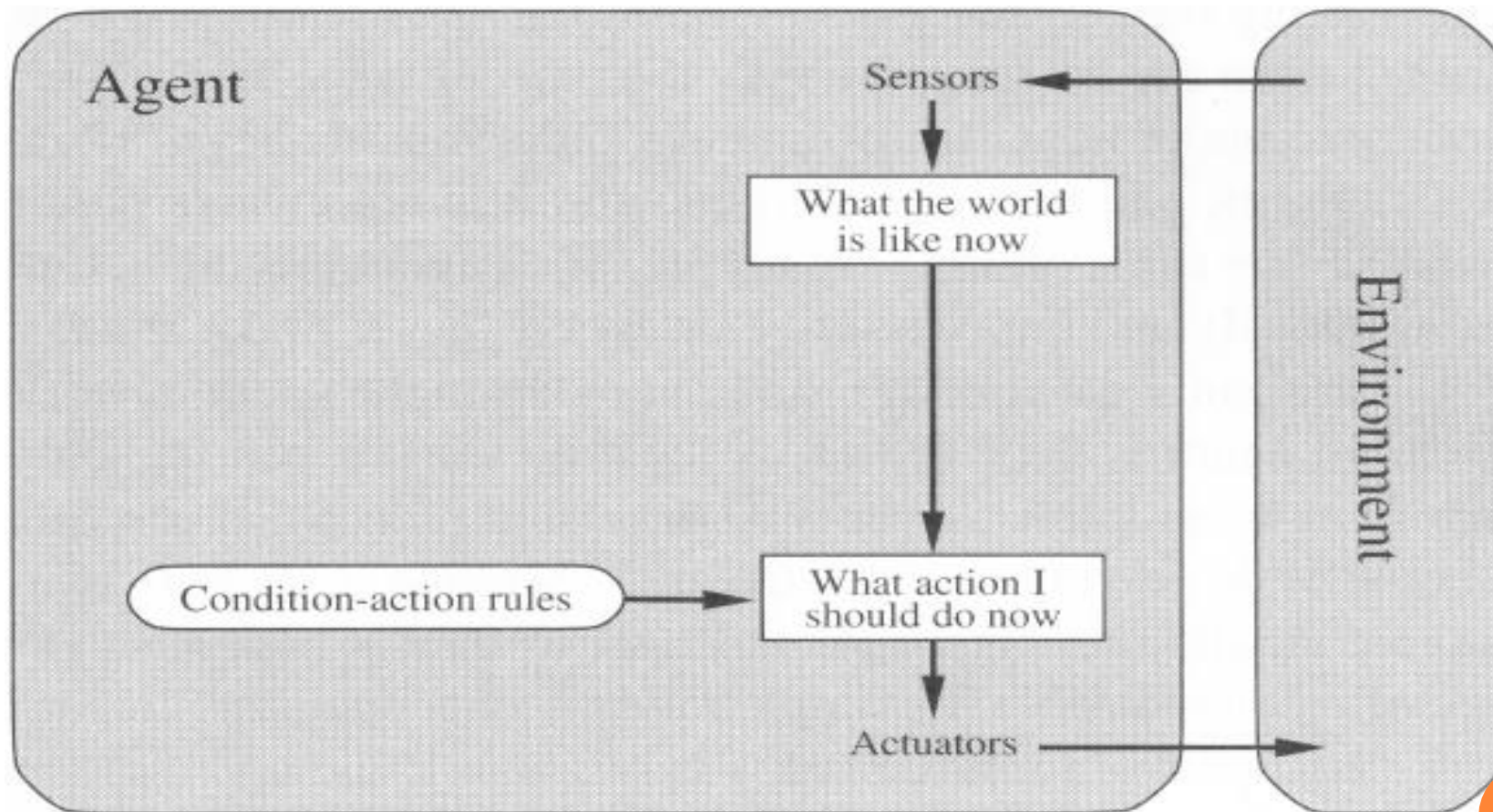
TYPES OF AGENT PROGRAMS

✧ The code-based agent programs are of five kinds that embody principles underlying almost all the intelligent agents:

- Simple-reflex /Reactive agents
- Model-based reflex/ Deliberative agents
- Goal-based agents
- Utility-based agents
- Learning agents



SIMPLE-REFLEX AGENTS CONTD....



SIMPLE-REFLEX AGENTS

- It is the simplest kind of agent.
- The agent selects the action on the basis of the current percept and ignoring the rest of the percept history.
- It uses just ***condition-action rules***
 - The rules are like the form “if ... then ...”
- For instance, the simple-reflex program for a vacuum cleaner world is:

Function Reflex-Vacuum-Agent(*[location,status]*)

return an action

If *status* = *Dirty* **then return** *Suck*

else if *location* = *A* **then return** *Right*

else if *location* = *B* **then return** *left*



SIMPLE-REFLEX AGENTS CONTD....

- It is an efficient method as ignoring the percept sequence reduces the number of possibilities (for instance, 4^T to 4 in case of vacuum cleaner world).
- But the simple-reflex agents **have narrow range of applicability** because knowledge sometimes cannot be limited.
- It works only if the environment is **fully observable**.



SIMPLE-REFLEX AGENTS CONTD....

✚ **Function:** SIMPLE-REFLEX-AGENT (percept)
returns an action
static: *rules*- a set of condition-action rules.

state \leftarrow Interpret-Input (percept)
rule \leftarrow Rule-Match (*state*, *rules*)
action \leftarrow Rule-Action[*rule*]
return **action**



MODEL-BASED REFLEX AGENTS

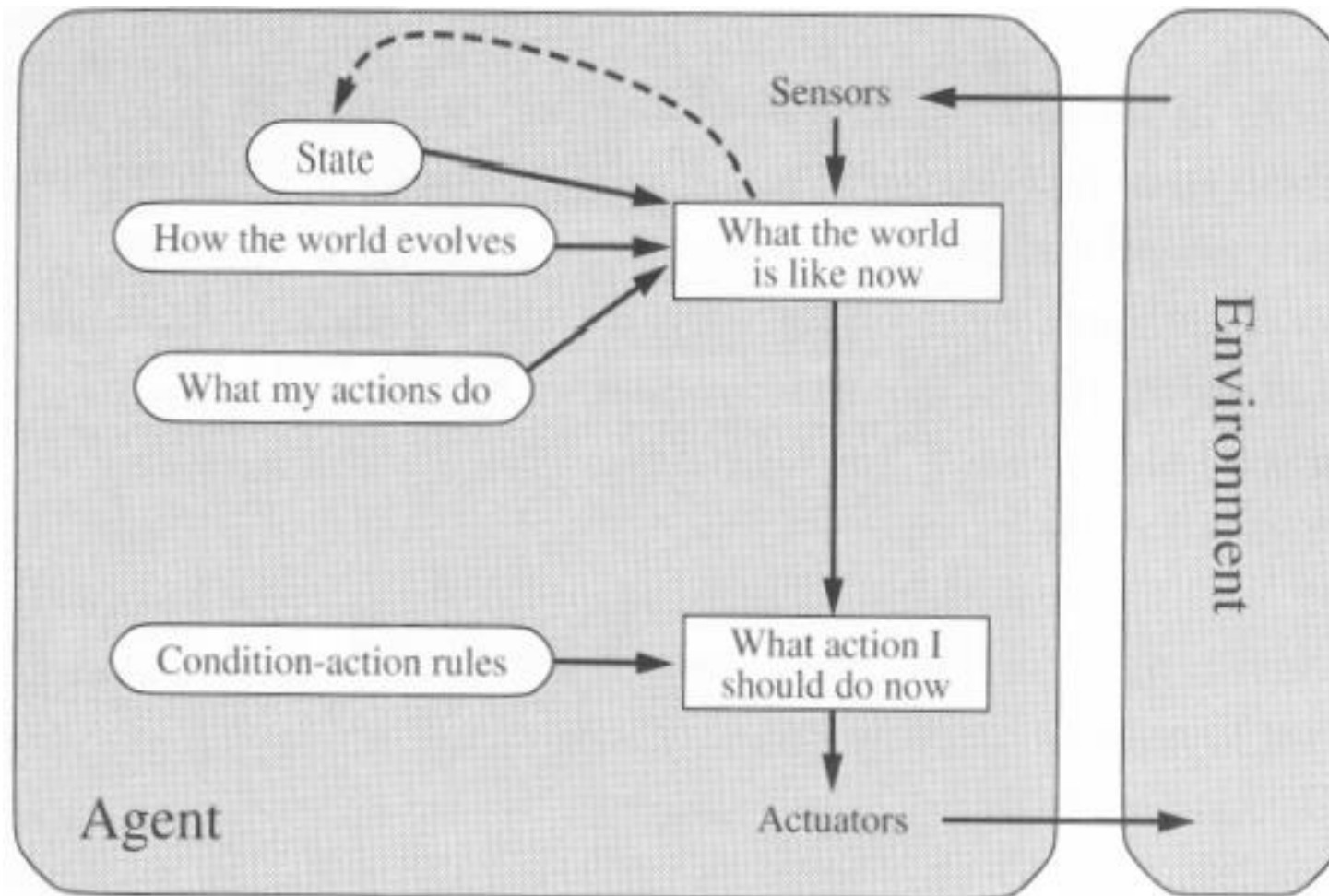
- ▢ For the world that is partially observable

E.g., driving a car and changing lane

- ▢ Requires two types of knowledge:
 - ▢ How the world evolves independently of the agent?
 - ▢ An overtaking car would be closer behind than it was a moment ago.
 - ▢ How the agent's actions affect the world?
 - ▢ When the agent turns the steering wheel clockwise, the car turns right.



MODEL-BASED REFLEX AGENTS CONTD....



MODEL-BASED REFLEX AGENTS CONTD....

Example Table Agent With Internal State

If	then
Saw an object ahead, and turned right, and it's now clear ahead	Go straight
Saw an object Ahead, turned right, and object ahead again	Halt
See no objects ahead	Go straight
See an object ahead	Turn randomly



MODEL-BASED REFLEX AGENTS CONTD....

✚ **Function:** RFLEX-AGENT-WITH-STATE (percept)
returns an action

static: *state*, a description of current world state

rules- a set of condition-action rules.

action- the most recent action, initially none

state \leftarrow Update-state(*state*, *action*, *percept*)

rule \leftarrow Rule-Match (*state*, *rules*)

action \leftarrow Rule-Action[*rule*]

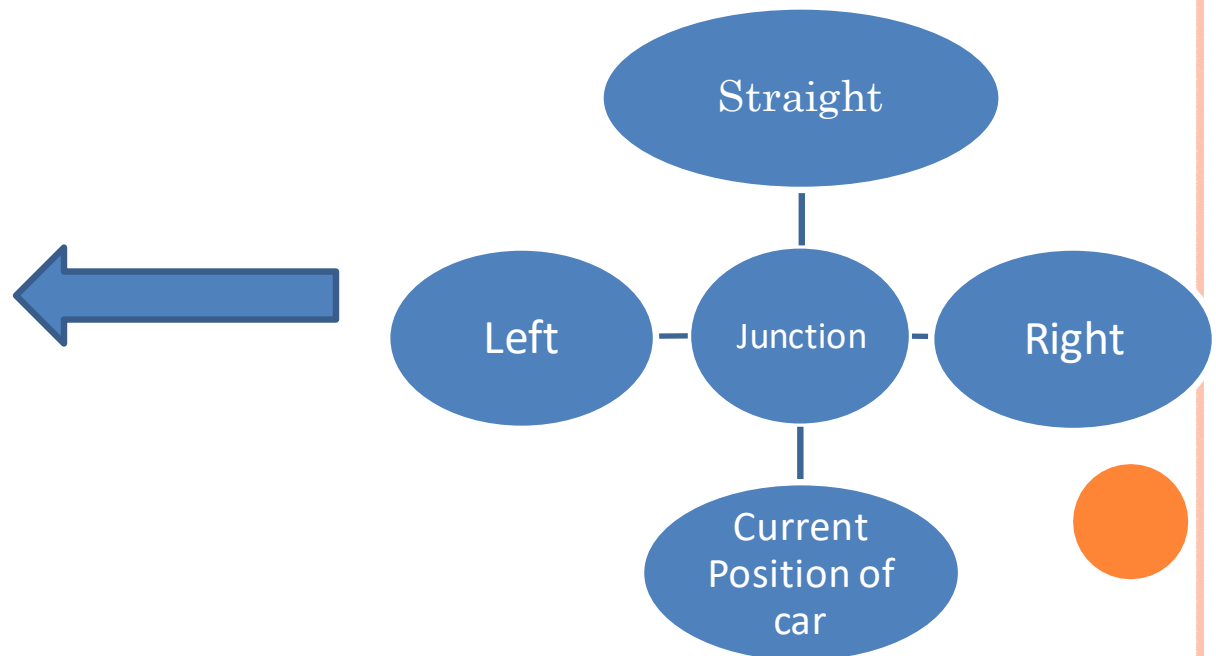
return **action**



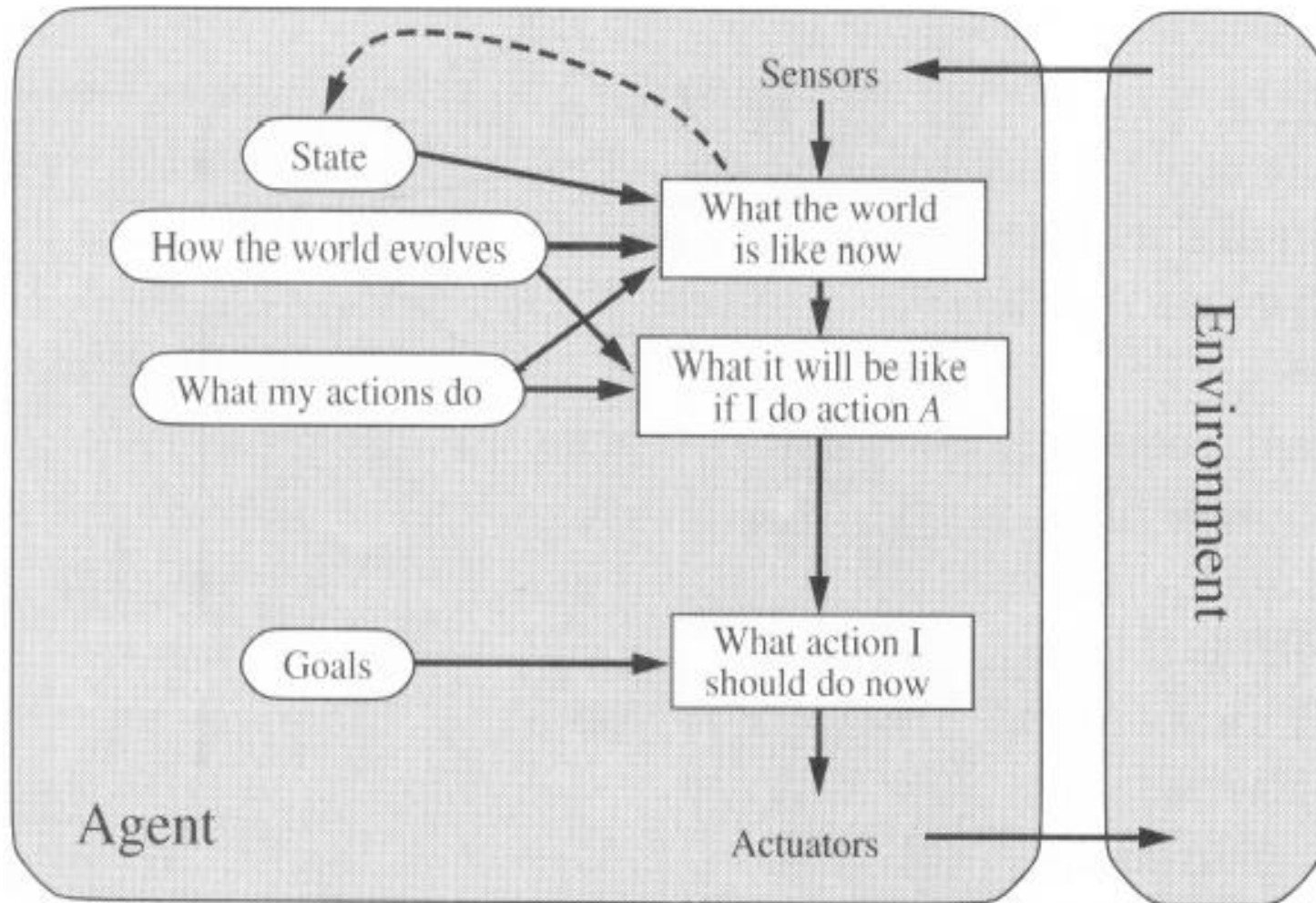
GOAL-BASED AGENTS

- ✧ The knowledge of the current state of the environment is not always sufficient.
- ✧ In addition to the current state, the agent needs some sort of **goal** information that describes situations that are desirable.
- ✧ For example,

**Passenger's
Destination**



GOAL-BASED AGENTS CONTD....




GOAL-BASED AGENTS CONTD....

- ⌘ Sometimes goal-based action selection is straight forward as there is single action that leads to a goal.
- ⌘ Sometimes it will be more tricky, when the agent needs to consider long sequence of actions to find a way to achieve the goal.
- ⌘ In such situations,
Searching (searching the set of actions) and
Planning (way the actions are executed)



GOAL-BASED AGENTS CONTD....

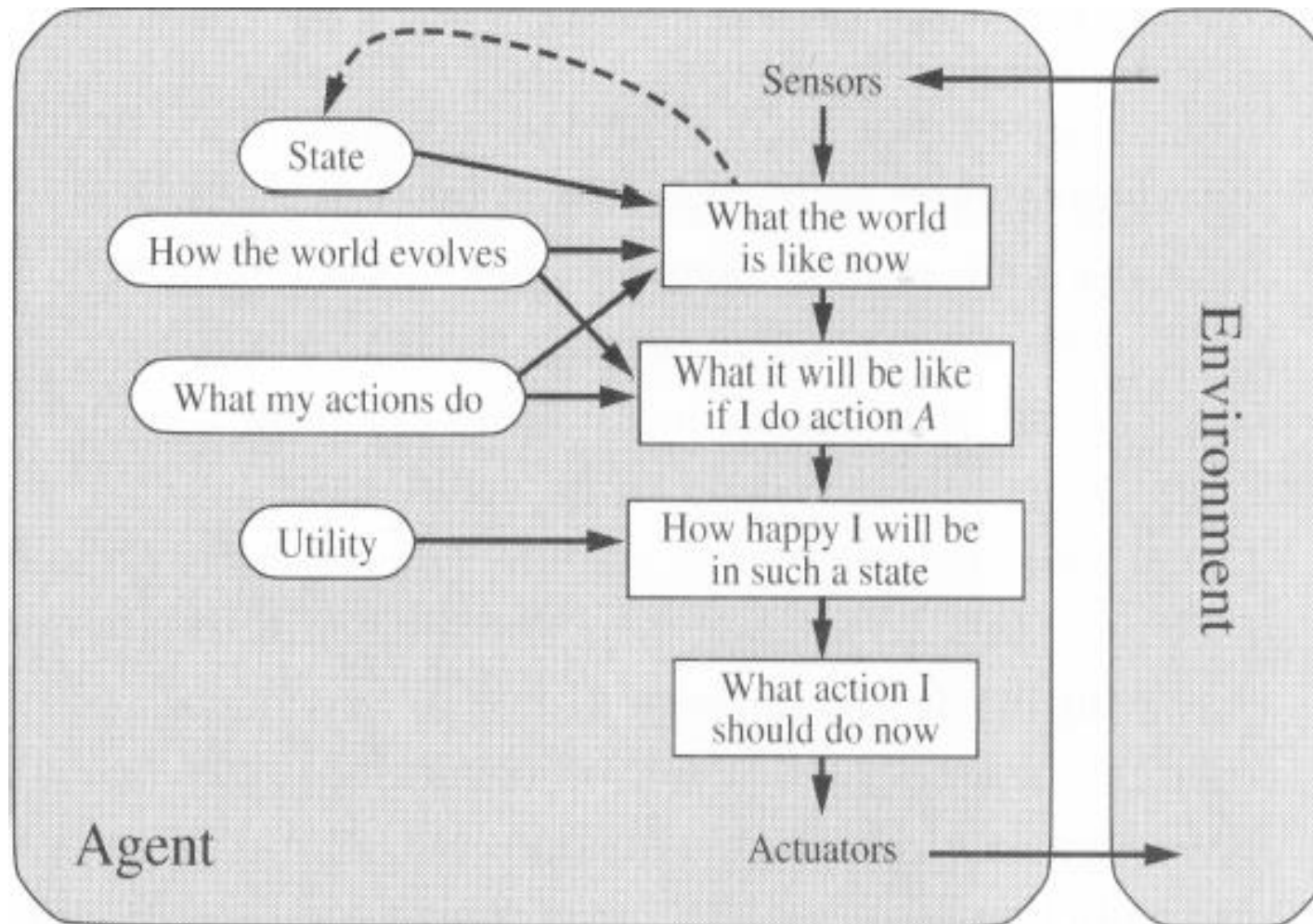
- ✧ In the reflex agents design, the information of both concepts is not explicitly represented.
 - ✧ For instance, the reflex agent brakes when it sees brakes light whereas the goal –based agent explains that the only action that will achieve the goal is not hitting the other cars.
 - ✧ In goal-based agents, the knowledge that support its decision is explicitly represented and can be modified.
 - ✧ For instance, if it starts to rain, the agent can update its knowledge of how effectively its brakes will operate; this automatically cause all of the relevant behavior to be altered whereas in the reflex agents we have to re-write many if-else conditions.
- 

UTILITY-BASED AGENTS

- ✧ Goals alone are not enough to generate high quality information.
- ✧ There are **many action sequences** that can satisfy the goal.
- ✧ Some of the action are better and some are worse. So, we need to measure the **utility** for the agent to decide the actions to choose.
- ✧ If goal measures success, then utility measures the **degree of success** (how successful it is).
- ✧ **Utility** is therefore a **function**, that maps a state to a real number which describes the **degree of happiness** (success).



UTILITY-BASED AGENTS CONTD....



UTILITY-BASED AGENTS CONTD....

▢ Utility has several advantages:

- When there are conflicting goals (safety and speed),
 - ▢ Only some of the goals but not all can be achieved
 - ▢ utility describes the appropriate trade-off
- When there are several goals and none of them can be achieved certainly.
 - ▢ utility provides a way for the decision-making

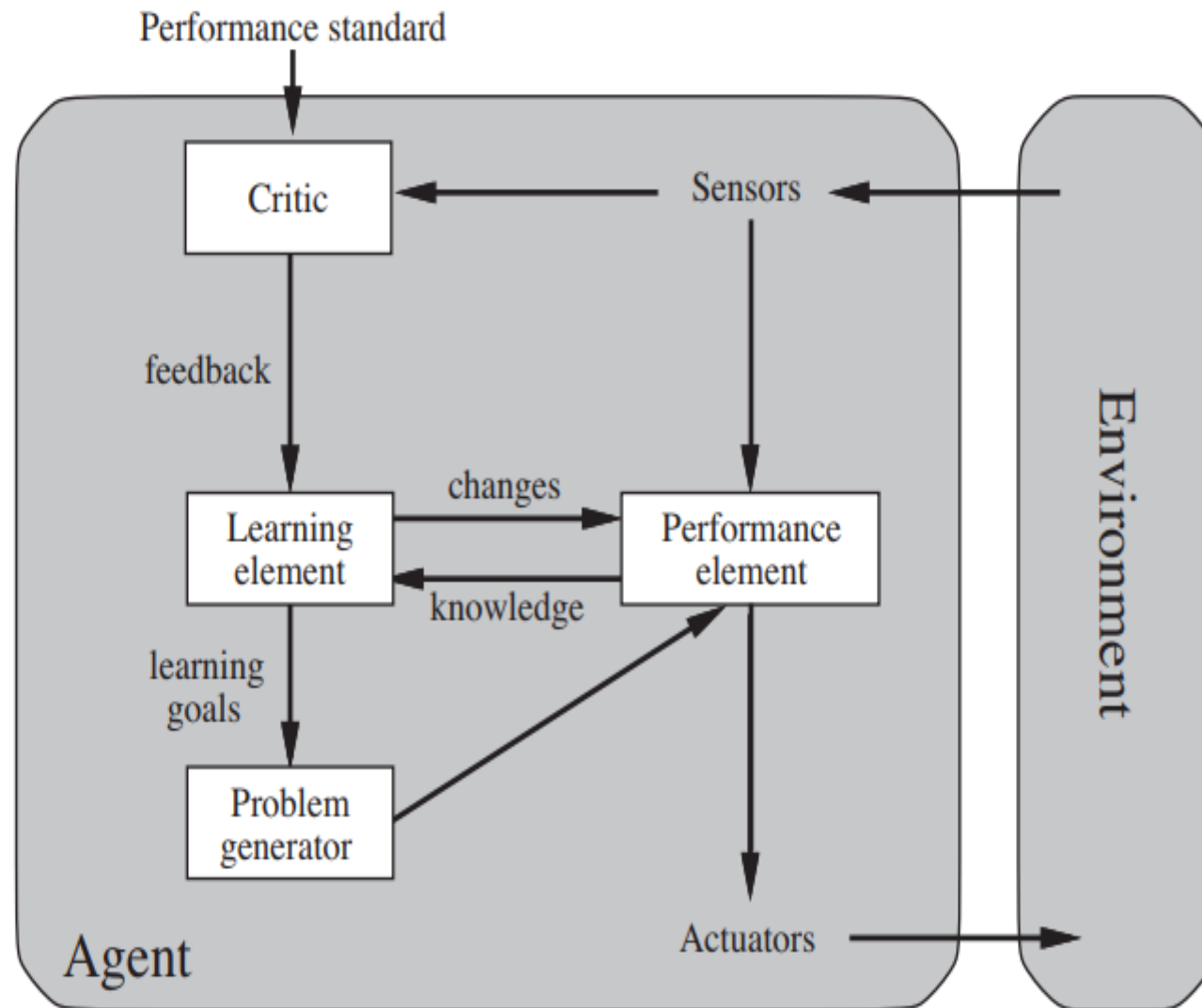


LEARNING AGENTS

- A learning agent in AI is the type of agent which can learn from its past experiences, or it has learning capabilities.
- It starts to act with basic knowledge and then able to act and adapt automatically through learning.



LEARNING AGENTS CONTD....



LEARNING AGENTS CONTD....

- A learning agent has mainly four conceptual components, which are:
 - **Learning element:** It is responsible for making improvements by learning from environment.
 - **Critic:** Learning element takes feedback from critic which describes that how well the agent is doing with respect to a fixed performance standard.
 - **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.

