AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH (AIUB)
FACULTY OF SCIENCE & INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

CSC 4121 A.I & EXPERT SYSTEM

TITLE |	Implementation of Intelligent Agents
PREREQUISITE |	Knowledge of Arrays| pseudo-code | Programming
OBJECTIVE |	To examine nature agents, environments, and the coupling between them.

THEORY | Rationality | Basic Kinds of Agents

Intelligent Agent

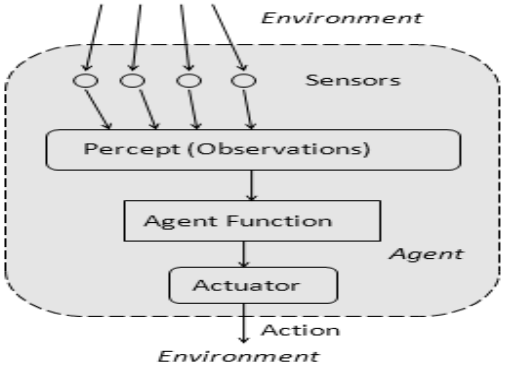
- Agent: entity in a program or environment capable of generating action.
- An agent uses perception of the environment to make decisions about actions to take.
- The perception capability is usually called a sensor.
- The actions can depend on the most recent perception or on the entire history (percept sequence).

Agent Function

- The agent function is a mathematical function that maps a sequence of perceptions into action.
- The function is implemented as the agent program.
- The part of the agent taking an action is called an actuator.
- environment -> sensors -> agent function -> actuators -> environment

Rational Agent

- A rational agent is one that can take the right decision in every situation.
- Performance measure: a set of criteria/test bed for the success of the agent's behavior.
- The performance measures should be based on the desired effect of the agent on the environment.



Rationality The agent's rational behavior depends on:

- the performance measure that defines success
- the agent's knowledge of the environment
- the action that it can perform
- the current sequence of perceptions.
- Definition: for every possible percept sequence, the agent is expected to take an action that will maximize its performance measure.

Agent vs. Program

- Size - an agent is usually smaller than a program.
- Purpose - an agent has a specific purpose while programs are multi-functional.
- Persistence - an agent's life span is not entirely dependent on a user launching and quitting it.
- Autonomy - an agent doesn't need the user's input to function.

Simple Agents

Table-driven agents:

- The function consists in a lookup table of actions to be taken for every possible state of the environment.
- If the environment has n variables, each with t possible states, then the table size is tn.
- Only works for a small number of possible states for the environment.

```
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.

Implementation Issue:

```
percepts = []                                  # create a list(array) of possible percepts
table = {}                                     # crate (key, value) pairs of percept   sequence

def table_agent (percept):                     # define a function with parameter percept
    action = True                              #Boolean variable action
    percepts.append(percept)                   # append new percept in the percepts list defined before
    action = lookup(percepts, table)           # use a function name lookup() to return action
    return action
```

Lab Task 1: Consider the above pseudocode for table driven agent program and implement it for the following vacuum cleaning agent environment and look up table, using any programming language you prefer.

```
loc_A, loc_B =
(0, 0), (1, 0)
# The two locations
for the Vacuum world
```

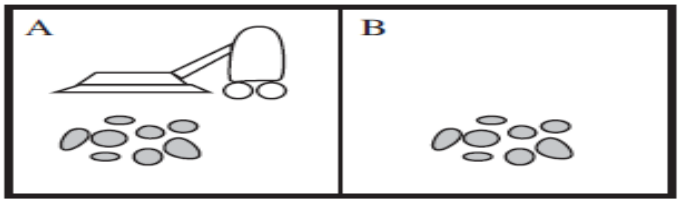


Figure 2.2 A vacuum-cleaner world with just two locations.

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
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Simple reflex agents:

- Deciding on the action to take based only on the current perception and not on the history of perceptions.
- Based on the condition-action rule: (if (condition) action)
- Works if the environment is fully observable

```
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.

Lab Task 2: Consider the above pseudocode for simple reflex agent program and implement it for the same vacuum cleaning agent environment used in lab task1, using any programming language you prefer.

```
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.

Model-Based Reflex Agents

- If the world is not fully observable, the agent must remember observations about the parts of the environment it cannot currently observe.
- This usually requires an internal representation of the world (or internal state).
- Since this representation is a model of the world, we call this model-based agent.

Lab Task 3: Consider the pseudocode for Model based reflex agent program and implement it for the same vacuum cleaning agent environment used in lab task1, using any programming language you prefer.

Model-Based Vacuum Cleaning Reflex Agents:

- An agent that keeps track of what locations are clean or dirty.
- Same as Reflex Vacuum Agent, except if everything is clean, do No Operation.

```
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.