OBJECTIVES:

- To know about how to implement an agent.

BACKGROUND STUDY

- Should have prior knowledge on any programming language to implement the agent.
- Should have knowledge on agent, simple reflex agent and vacuum cleaner agent that are covered in theory class.
- Need knowledge on function, 2D array, etc.

RECOMMENDED READING

- Artificial Intelligence: A Modern Approach – By Stuart Russel and Peter Norvig.

Vacuum-Cleaner Agent

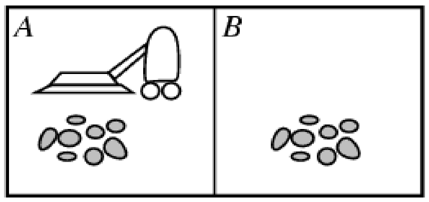
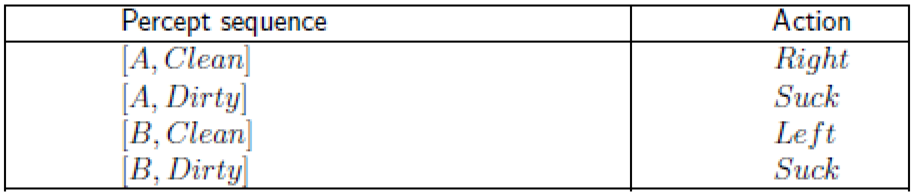


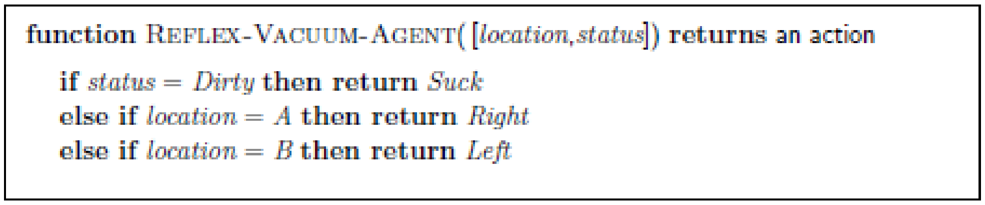
Figure 1: Vacuum-cleaner world

The vacuum-cleaner 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. A partial tabulation of this agent function is shown in Figure-2 and an agent program that implements it appears in Figure-3.



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

Figure 2: Vacuum-cleaner agent rules table



```
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
```

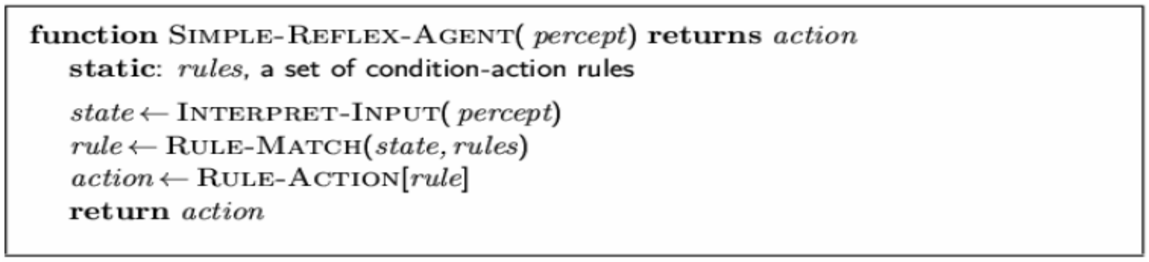
Figure 3: Reflex-vacuum-agent algorithm

Simple Reflex Agent

The simplest kind of agent is the **simple reflex agent.** 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-2 is a simple reflex agent, because its decision is based only on the current location and on whether that location contains dirt. Art agent program for this agent is shown in Figure-3.

The program in Figure-3 is specific to one particular vacuum environment. A more general and flexible approach is first to build a general-purpose interpreter for condition-action rules and then to create rule sets for specific task environments.

The agent program, which is also very simple, is shown in Figure-4. The INTERPRET-INPUT function generates an abstracted description of the current slate from the percept, and the RULE-MATCH function returns the first rule in the set of rules that matches the given state description. Note that the description in terms of "rules" and "matching" is purely conceptual; actual implementations can be as simple as a collection of logic gates implementing a Boolean circuit.



```
function SIMPLE-REFLEX-AGENT( percept) returns action
    static: rules, a set of condition-action rules

    state ← INTERPRET-INPUT( percept)
    rule ← RULE-MATCH(state, rules)
    action ← RULE-ACTION[rule]
    return action
```

Figure 4: Simple-reflex-agent algorithm

LABROTORY EXERCISES

1. Write a program to design a vacuum-cleaner agent which is a simple-reflex agent using the algorithm shown in Figure-3.

   Input: vacuum cleaner’s location information *location* and dirt information *status*
   Output: agent’s action (left, right, suck)

2. Write a program to design a vacuum-cleaner agent which is a simple-reflex agent using the algorithm shown in Figure-4.

   Input: vacuum cleaner’s location information *location* and dirt information *status*
   Output: agent’s action (left, right, suck)