**Categories of intelligent agent**

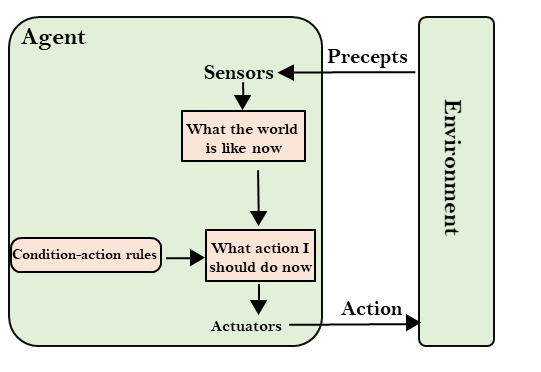
**Agent:**

Agents can be grouped into five classes based on their degree of perceived intelligence and capability. All these agents can improve their performance and generate better action over the time. These are given below:

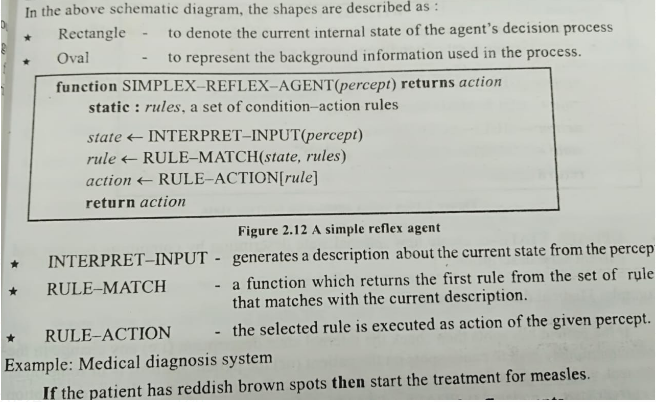
* Simple Reflex Agent
* Model-based reflex agent
* Goal-based agents
* Utility-based agent
* Learning agent

**Simple Reflex agent:**

* The Simple reflex agents are the simplest agents. These agents take decisions on the basis of the current percepts and ignore the rest of the percept history.
* These agents only succeed in the fully observable environment.
* The Simple reflex agent does not consider any part of percepts history during their decision and action process.
* The Simple reflex agent works on Condition-action rule, which means it maps the current state to action. Such as a Room Cleaner agent, it works only if there is dirt in the room.
* Problems for the simple reflex agent design approach:
  + They have very limited intelligence
  + They do not have knowledge of non-perceptual parts of the current state
  + Mostly too big to generate and to store.
  + Not adaptive to changes in the environment.



This agent selects actions based on the agents current perception or the world and



**2. Water jug problem**

### Water-Jug Problem

**Problem:**You are given two jugs, a 4-gallon one and a 3-gallon one.Neither has any measuring mark on it.There is a pump that can be used to fill the jugs with water.How can you get exactly 2 gallons of water into the 4-gallon jug.

**Solution:**

The state space for this problem can be described as the set of ordered pairs of integers **(x,y)**

Where,

X represents the quantity of  water in the 4-gallon jug  **X= 0,1,2,3,4**

Y represents the quantity of water in 3-gallon jug **Y=0,1,2,3**

**Start State: (0,0)**

**Goal State: (2,0)**

Generate production rules for the water jug problem

**Production Rules:**

|  |  |  |
| --- | --- | --- |
| **Rule** | **State** | **Process** |
| 1 | (X,Y | X<4) | (4,Y)  {Fill 4-gallon jug} |
| 2 | (X,Y |Y<3) | (X,3)  {Fill 3-gallon jug} |
| 3 | (X,Y |X>0) | (0,Y)  {Empty 4-gallon jug} |
| 4 | (X,Y | Y>0) | (X,0)  {Empty 3-gallon jug} |
| 5 | (X,Y | X+Y>=4 ^ Y>0) | (4,Y-(4-X))  {Pour water from 3-gallon jug into 4-gallon jug until 4-gallon jug is full} |
| 6 | (X,Y | X+Y>=3 ^X>0) | (X-(3-Y),3)  {Pour water from 4-gallon jug into 3-gallon jug until 3-gallon jug is full} |
| 7 | (X,Y | X+Y<=4 ^Y>0) | (X+Y,0)  {Pour all water from 3-gallon jug into 4-gallon jug} |
| 8 | (X,Y | X+Y <=3^ X>0) | (0,X+Y)  {Pour all water from 4-gallon jug into 3-gallon jug} |
| 9 | (0,2) | (2,0)  {Pour 2 gallon water from 3 gallon jug into 4 gallon jug} |

**Initialization:**

Start State: (0,0)

Apply Rule 2:

|  |  |
| --- | --- |
| (X,Y | Y<3)    -> | (X,3)  {Fill 3-gallon jug} |

Now the state is **(X,3)**

**Iteration 1:**

Current State: **(X,3)**

Apply Rule 7:

|  |  |
| --- | --- |
| (X,Y | X+Y<=4 ^Y>0) | (X+Y,0)  {Pour all water from 3-gallon jug into 4-gallon jug} |

Now the state is **(3,0)**

**Iteration 2:**

Current State : **(3,0)**

Apply Rule 2:

|  |  |
| --- | --- |
| (X,Y | Y<3)    -> | (3,3)  {Fill 3-gallon jug} |

Now the state is **(3,3)**

**Iteration 3:**

Current State:**(3,3)**

Apply Rule 5:

|  |  |
| --- | --- |
| (X,Y | X+Y>=4 ^ Y>0) | (4,Y-(4-X))  {Pour water from 3-gallon jug into 4-gallon jug until 4-gallon jug is full} |

Now the state is **(4,2)**

**Iteration 4:**

Current State : (4,2)

Apply Rule 3:

|  |  |
| --- | --- |
| (X,Y | X>0) | (0,Y)  {Empty 4-gallon jug} |

Now state is (0,2)

**Iteration 5:**

Current State : (0,2)

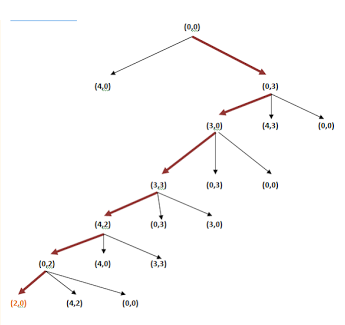
Apply Rule 9:

|  |  |
| --- | --- |
| (0,2) | (2,0)  {Pour 2 gallon water from 3 gallon jug into 4 gallon jug} |

Now the state is **(2,0)**

**Goal Achieved.**

**State Space Tree:**

****