National University of Computer & Emerging Sciences

Lab Manual
CS461: Artificial Intelligence Lab

Course Instructor	Dr. Hafeez-Ur-Rehman
Lab Instructor	Muhammad Yousaf
Semester	Spring 2021

Lab # 03 – Agents & Environments

Outline

1. Agents
2. Agent programs
3. Rationality
4. Environments
5. Agent structures
6. Multi-agent systems

Agents

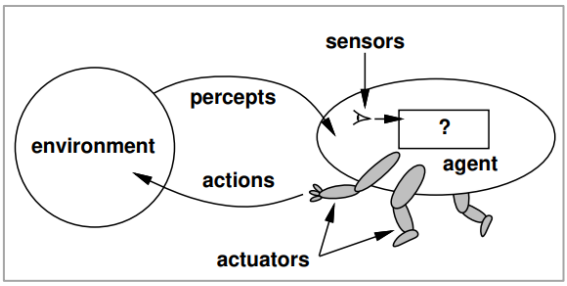
An agent is an entity that perceives and acts in an environment
Agents include

- animal agents
- human agents
- robotic agents (robots)
- software agents (softbots)
  - internet agents
  - crawler
  - webbot
  - email agent
  - search agent, etc.
  - chatbots
  - Cortana/Siri/GAssistant/Waston/Alexa/FMessenger/· · ·

Single agent or usually multi-agents (so-called distributed AI)

Agents and Environments

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

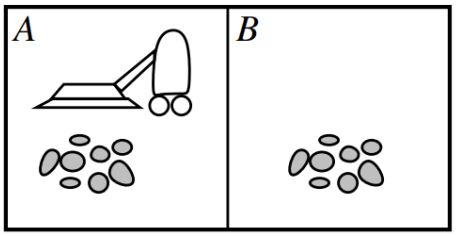


Sensors and Actuators

A sensor measures some aspect of the environment in a form
that can be used as input by an agent
– vision, hearing, touch
– radio, infrared, GPS, wireless signals
– active sensing: send out a signal (such as radar or ultrasound) and sense the reflection of this signal off of the environment i.e. IoT (Internet of Things)

Perception provides agents with information about the world they inhabit by interpreting the response of sensors.
Actuators
– hands, legs, vocal tract etc.
– automated taxi: those available to a human driver
e.g., accelerator, steering, braking and so on

Example: Vacuum-cleaner world & Agent



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

Example: Robot cleaner, say, iRobot Roombat

Percepts: location and contents, e.g., [A, Dirty]
Actions: Lef t, Right, Suck, NoOp

Agent programs

The agent program runs on the physical architecture to produce the agent function
agent = architecture + program
program = algorithm + data
The agent program takes a single percept as input, keeps internal state

```
function SKELETON-AGENT(percept) returns action // output result
  inputs: percept, input from sensors  // may be omitted
  persistent: memory, the agent's memory of the world
  memory ← UPDATE-MEMORY(memory, percept)
  action ← CHOOSE-BEST-ACTION(memory)
  memory ← UPDATE-MEMORY(memory, action)
  return action  // output
```

```
import environment as en
import domain as do
import action as ac

def skeleton_agent(percept):
    m = []
    m = do.push(percept)
    a = ac.choose()
    #chosen from action class
    m = ac.update()
    return a
```

Example: A Vacuum-cleaner Agent

```
def TABLE-DRIVEN-AGENT(percept)
  persistent: percepts, a sequence, initially empty
        table, a table, indexed by percept sequences, initially fully specified
  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

```
def Table_Driven_Vacuum_Agent(table):
    """

    A table is provided as a dictionary of all
    {percept_sequence:action} pairs.
    """
    percepts = []

    def agent(percept):
        percepts.append(percept)
        action = table.get(tuple(percepts))
        return action

    return agent
```

Python: A Vacuum-cleaner Agent

```
def Table_Driven_Vacuum_Agent():
    """Tabular approach towards vacuum world

    >>> agent = Table_Driven_Vacuum_Agent()
    >>> environment = VacuumEnvironment()
    >>> environment.add_thing(agent)
    >>> environment.run()
    >>> environment.status == {(1,0):'Clean' , (0,0) : 'Clean'}
    True
    """

    table = {((loc_A, 'Clean'),): 'Right',
             ((loc_A, 'Dirty'),): 'Suck',
             ((loc_B, 'Clean'),): 'Left',
             ((loc_B, 'Dirty'),): 'Suck',
             ((loc_A, 'Dirty'), (loc_A, 'Clean')): 'Right',
             ((loc_A, 'Clean'), (loc_B, 'Dirty')): 'Suck',
             ((loc_B, 'Clean'), (loc_A, 'Dirty')): 'Suck',
             ((loc_B, 'Dirty'), (loc_B, 'Clean')): 'Left',
             ((loc_A, 'Dirty'), (loc_A, 'Clean'), (loc_B, 'Dirty')): 'Suck',
             ((loc_B, 'Dirty'), (loc_B, 'Clean'), (loc_A, 'Dirty')): 'Suck'}
    return Agent(Table_Driven_Vacuum_Agent(table))
```

Rationality

A rational agent is one that does right thing – to achieve the best performance

"Goals" specifiable by performance measure defining a numerical value for any environment history
Rational action: whichever action maximizes the expected value of the performance measure given the percept sequence to date

PEAS (Performance/Environment/Actuators/Sensors)

To design a rational agent, we must specify the task environment
E.g., an automated taxi (intelligent vehicle):

- Performance measure??
- Environment??
- Actuators??
- Sensors??

Example: Automated Taxi Agent

To design a rational agent, we must specify the task environment
E.g., intelligent vehicle (an automated taxi):
Performance measure?? safety, destination, profits, legality, . . .
Environment?? streets, traffic, pedestrians, weather, . . .
Actuators?? steering, accelerator, brake, horn, speaker/display, . . .
Sensors?? video, accelerometers, gauges, engine sensors, GPS, . . .

Example: Internet Shopping Agent

Performance measure?? price, quality, appropriateness, efficiency, . . .
Environment?? web sites, vendors, shippers, . . .
Actuators?? display to user, follow URL, fill in form, . . .
Sensors?? pages (text, graphics, scripts), . . .

Environments

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??	Yes	Semi	Semi	No
Discrete??	Yes	Yes	Yes	No
Single-agent??	Yes	No	Yes (except auctions)	No

The environment type largely determines the agent design The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent.

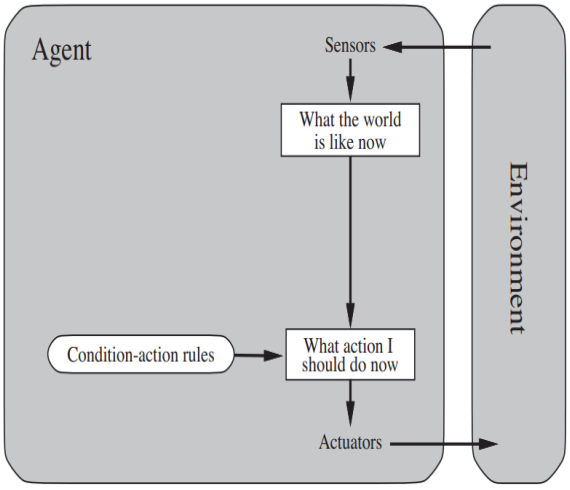
Agent structures

Four basic types in order of increasing generality:

1. simple reflex agents
2. model-based reflex agents
3. goal-based agents
4. utility-based agents

All these can be turned into – learning agents

Simple reflex agents

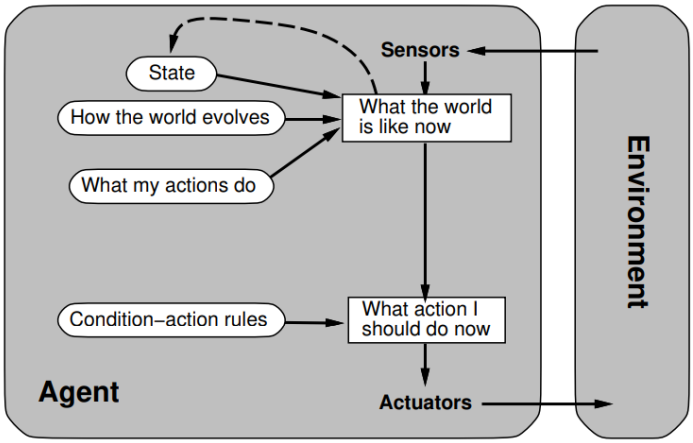



```
def SIMPLE-REFLEX-AGENT(percept)
persistent: rules, a set of condition-action rules
    state ← INTERPRET-INPUT(percept)
    rule ← RULE-MATCH(state, rules)
    action ← rule.ACTION
    return action
```

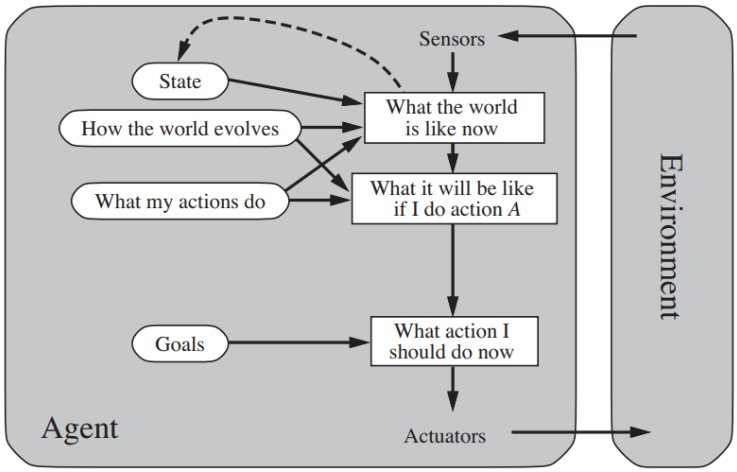
```
def REFLEX-VACUUM-AGENT([location, status])
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

Model-based reflex agents

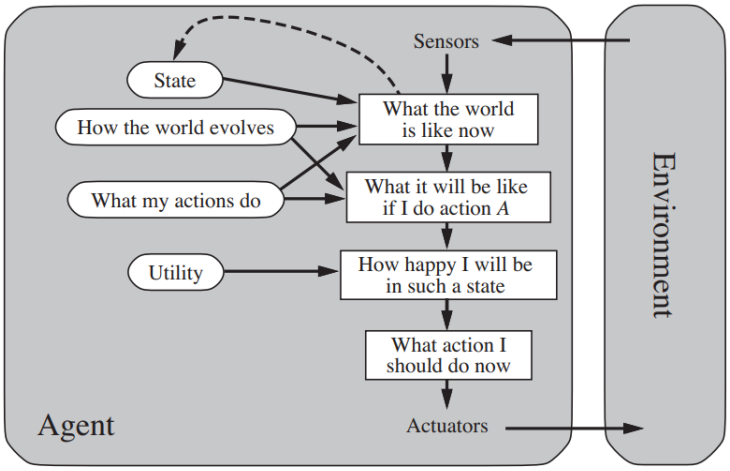
Reflex agents with (internal) state transition model - how the world works



Goal-based agents



Utility-based agents



Learning agents