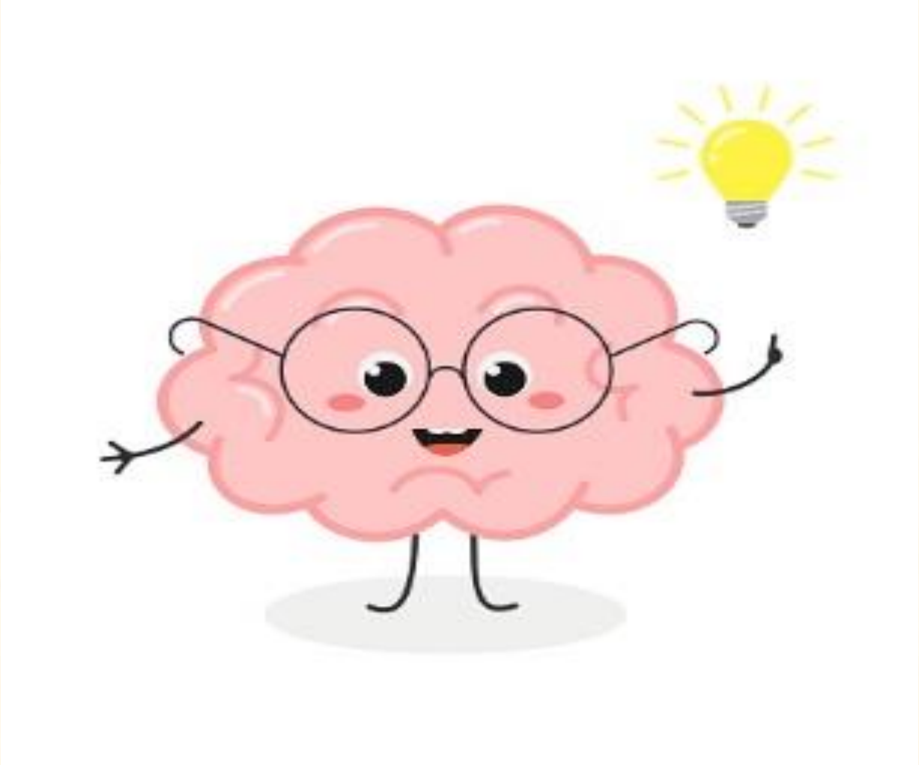
Chapter 2
Intelligent Agents



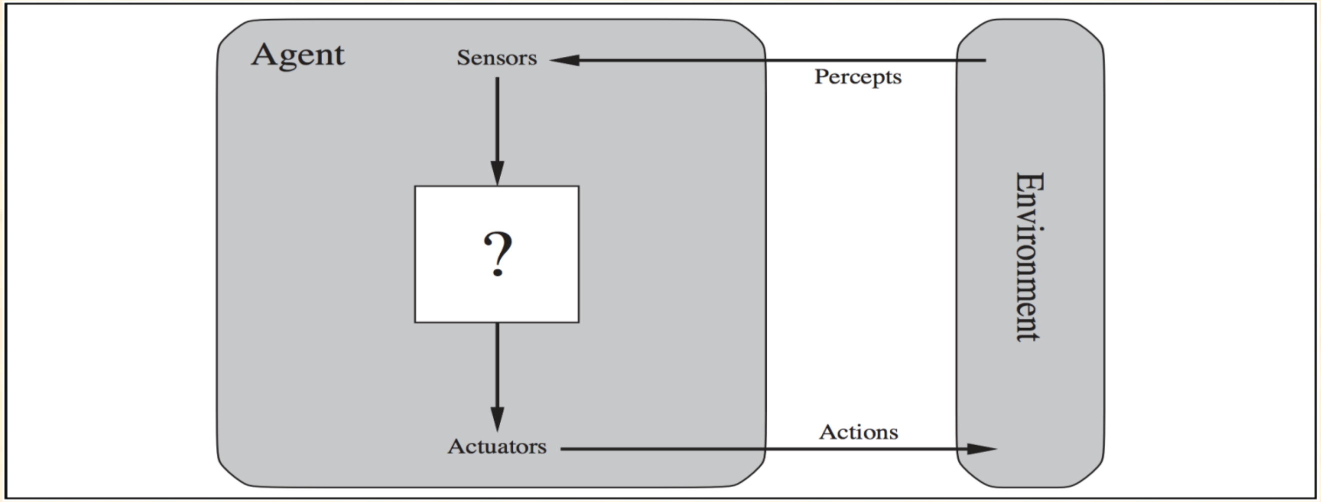
Source: Getty Images/iStockphoto

Outline

- Agents and environments
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

Agents

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



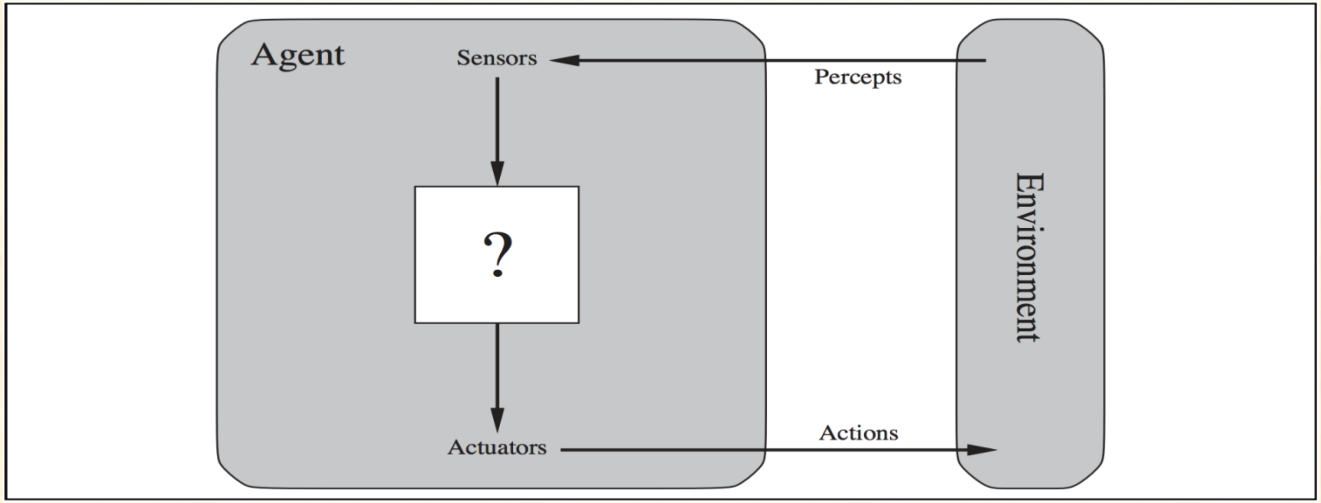
Human agent: sensors - eyes, ears, and other sensory organs; **actuators -** hands, legs, mouth etc

Robotic agent: sensors - cameras and infrared range finders ; **actuators -** various motors for actuators

Software agent: sensors - Key strokes, file contents, network packets
actuators - Displaying on screen, writing files, sending network packets

Agents

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



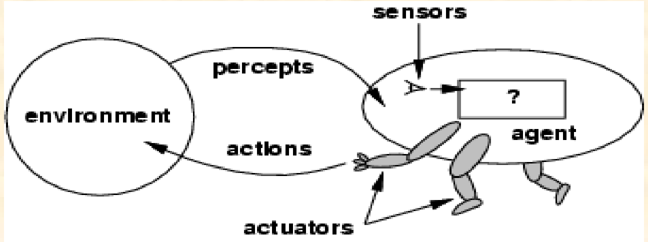
Percepts: agents perceptual inputs at any given instance
Percept sequence: complete history of agents perceived inputs
Agent functions: describes the agent's behavior

The agent function maps from percept histories to actions:

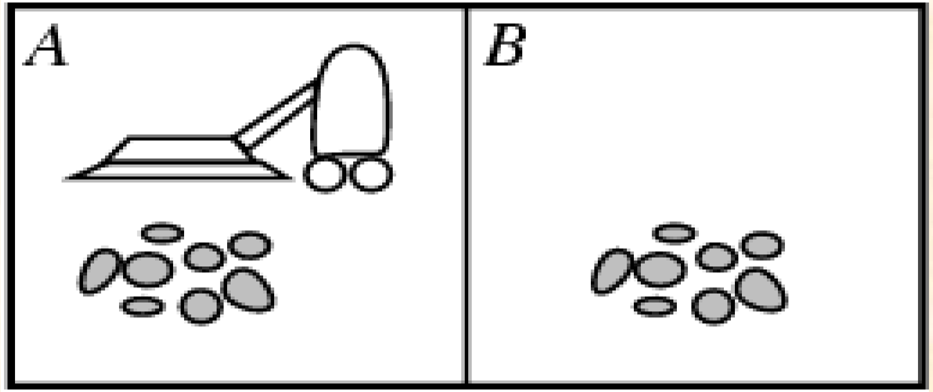
$$[f: \mathcal{P}^* \rightarrow \mathcal{A}]$$

The agent program runs on the physical architecture to produce *f*

agent = architecture + program

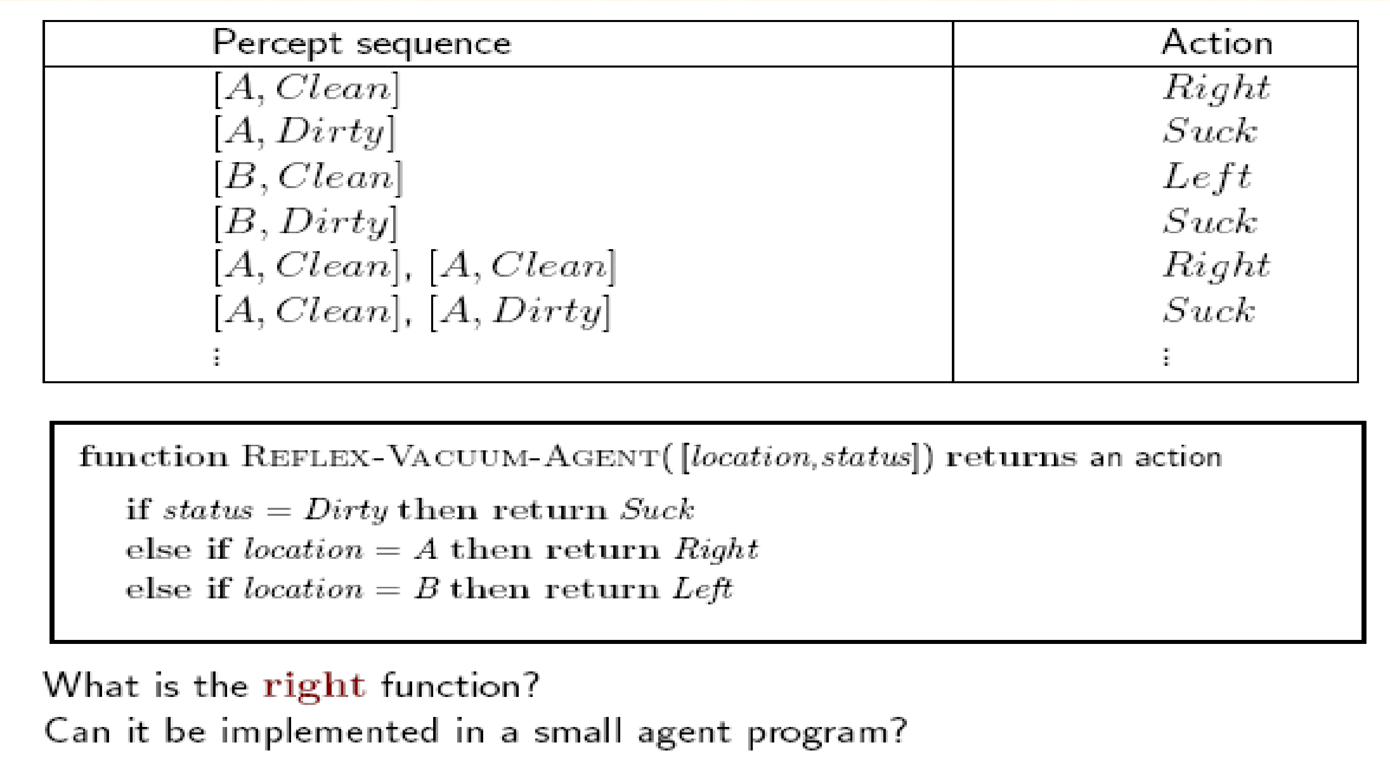


Vacuum-cleaner world



- Percepts: location and contents, e.g., [A, Dirty]
- Actions: *Left*, *Right*, *Suck*, *NoOp*

A vacuum-cleaner agent



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

```
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 $Left$
```

What is the right function?
Can it be implemented in a small agent program?

Good behavior: Concept of Rationality

Rational agent

- An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform.
- All entry in the table for the agent function are filled correctly.
- The right action is the one that will cause the agent to be most successful

How to measure success????

- Performance measure: A criterion for success of an agent's behavior
- E.g., performance measure of vacuum cleaner
  - amount of dirt cleaned up,
  - amount of time taken,
  - amount of electricity consumed,
  - amount of noise generated, etc.

Selection of performance measure:

- Average performance Vs energetic performance with long breaks
- Reckless life or safe but simple existence
- Economy with moderate poverty for all or some are super rich + some are very poor

Good behavior: Concept of Rationality

What is rational??

- The Performance measure that defines the criterion of success
- The agent’s prior knowledge of environment
- The actions that the agent performs
- The agent’s percept sequence to data

Def: ***For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.***

How to check if Vacuum cleaner is a rational agent?

- Performance measure: award one point for each clean square
- Environment: Geography of environment is known apriori – two squares A & B
- Actions: Left, Right, Such and NoOp
- Sensors: Agent perceives its location and checks if contains dirt or not

Under these circumstances Vacuum cleaner agent is rational.

Rational agents

- Agents can perform actions in order to modify future percepts so as to obtain useful information - information gathering, exploration

- An agent should learn from what it perceives

- An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt)

Nature of Environment

- **Building rational agents**
  - Task environments : “problems”
  - Rational agent : “solutions”

- **Specifying task environment**
  - PEAS: (Performance measure, Environment, Actuators, Sensors)
  - e.g., the task of designing an automated taxi driver:
    - **Performance measure**: Safe, fast, legal, comfortable trip, maximize profits
    - **Environment**: Roads, other traffic, pedestrians, customers
    - **Actuators**: Steering wheel, accelerator, brake, signal, horn
    - **Sensors**: Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard

PEAS

- Medical Diagnosis
- Part-picking robot

PEAS

- Agent: 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)

- Agent: 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

Environment types

- Fully observable vs. partially observable:
  - Fully observable - An agent's sensors give it access to the complete state of the environment at each point in time.
  - Sensors detect all aspects relevant to choice of action (performance measure)
  - Agent need not maintain any internal state to keep track of the world
  - Partially observable - due to noise or inaccurate sensors or parts of state are missing
  - E.g. taxi cannot what other drivers are thinking

- Deterministic vs. stochastic:
  - Deterministic - The next state of the environment is completely determined by the current state and the action executed by the agent.
  - E.g. taxi driver is stochastic, vacuum cleaner example is deterministic
  - (If the environment is deterministic except for the actions of other agents, then the environment is strategic)

- Episodic vs. sequential:
  - The agent's experience is divided into atomic "episodes"
  - each episode consists of the agent perceiving and then performing a single action, and the choice of action in each episode depends only on the episode itself.
  - In sequential environment, the current decision could affect all future decision
  - E.g chess playing , taxi driver – sequential
  - Fault detecting robot in assembly line - episodic

Environment types

- Static vs. dynamic:
  - The environment is changed while an agent is deliberating – dynamic else static.
  - Static – easy to deal, no importance to time
  - (The environment is semi dynamic if the environment itself does not change with the passage of time but the agent's performance score does)
  - E.g. taxi – dynamic, crossword puzzle – static, chess with clock - semidynamic
- Discrete vs. continuous:
  - Applied to state of environment, way time is handled, percepts/actions of agents
  - Discrete - A limited number of distinct, clearly defined percepts and actions.
  - E.g taxi – continuous ,
  - chess, crossword - discrete
- Single agent vs. multi agent:
  - Single - An agent operating by itself in an environment
  - E.g. crossword - single
  - Competitive multiagent – chess with opponent
  - Cooperative multiagent – taxi driving

Environment types

	Taxi driving	Chess playing with clock	Chess playing without clock
Observable	Partially	Fully	Fully
Deterministic	Stochastic	Strategic	Strategic
Episodic	Sequential	Sequential	Sequential
Static	Dynamic	Semi-dynamic	Dynamic
Discrete	Continuous	Discrete	Discrete
Agents	Multi	Single	Single

The real world is partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Agent functions and programs

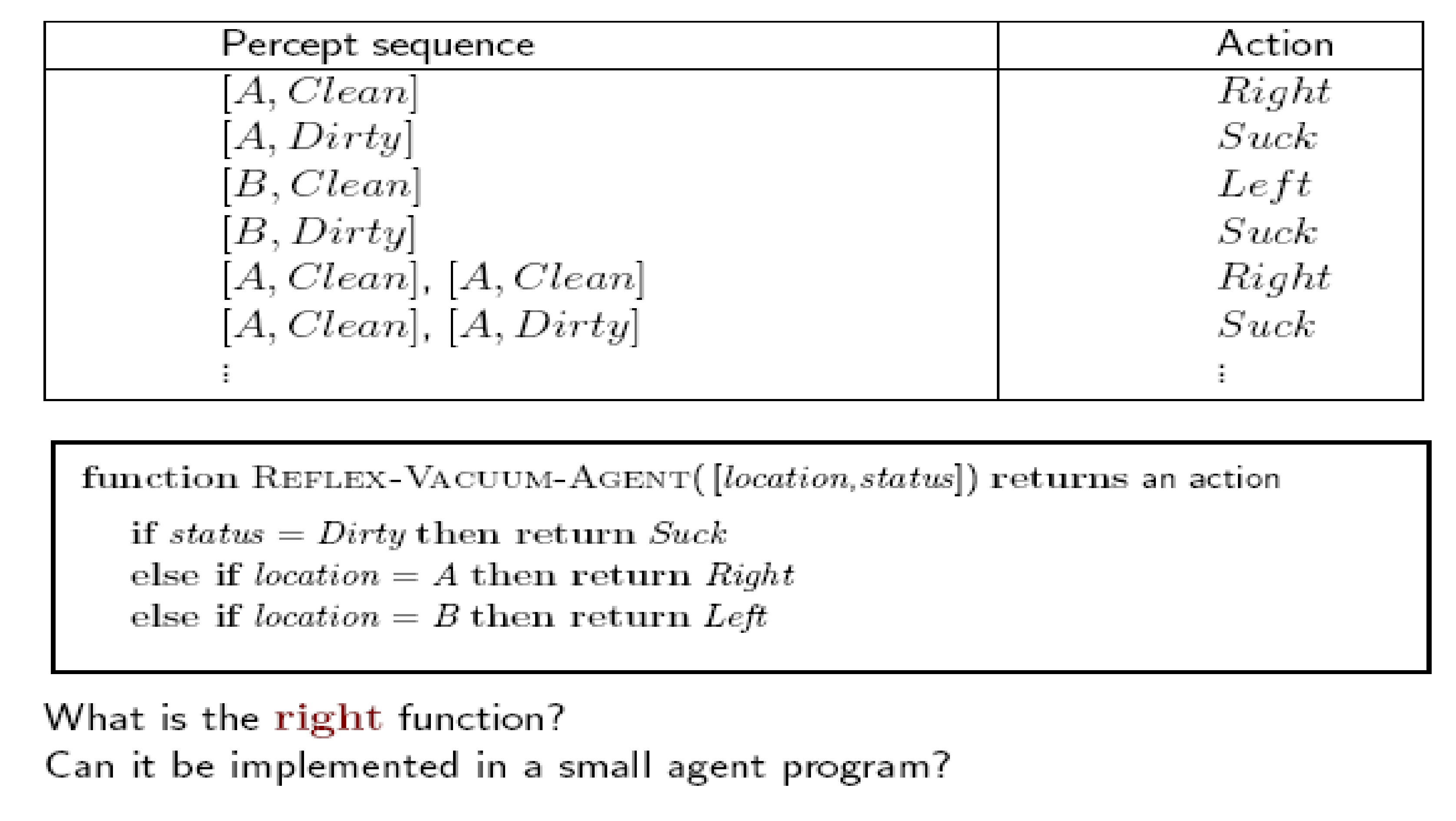
- An agent is completely specified by the agent function mapping percept sequences to actions
- One agent function (or a small equivalence class) is rational
- Aim: find a way to implement the rational agent function concisely

Table-lookup agent

action 🡐 LOOKUP(percepts, table)

- Drawbacks:
  - Huge table
  - Take a long time to build the table
  - No autonomy
  - Even with learning, need a long time to learn the table entries

Agent program for a vacuum-cleaner agent



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

```
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 $Left$
```

What is the right function?

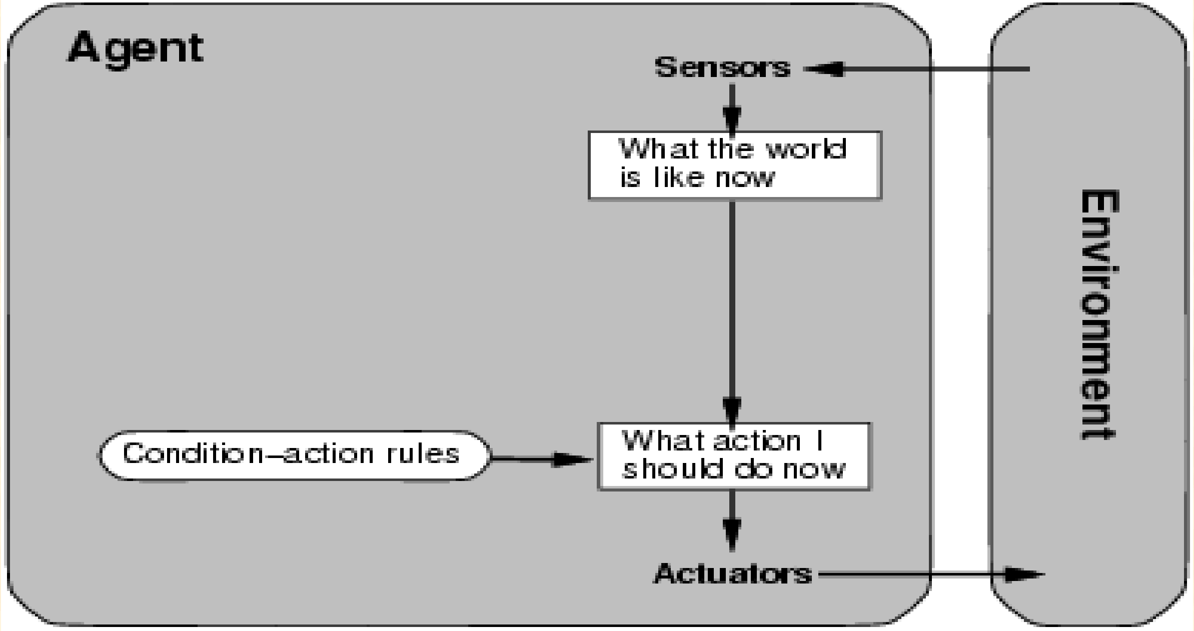
Can it be implemented in a small agent program?

Agent types

Four basic types in order of increasing generality:

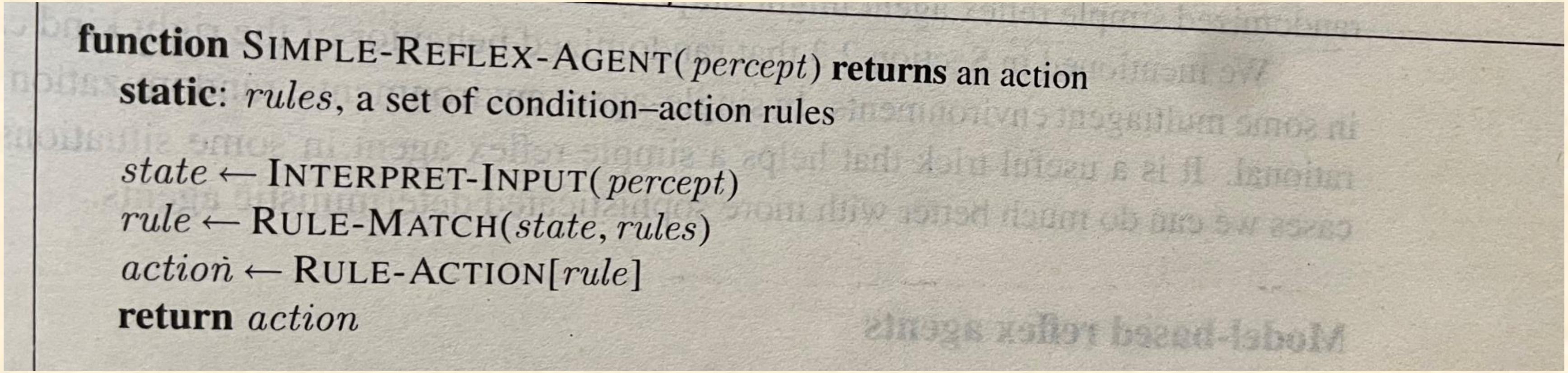
- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents
- Learning agents

Simple reflex agents



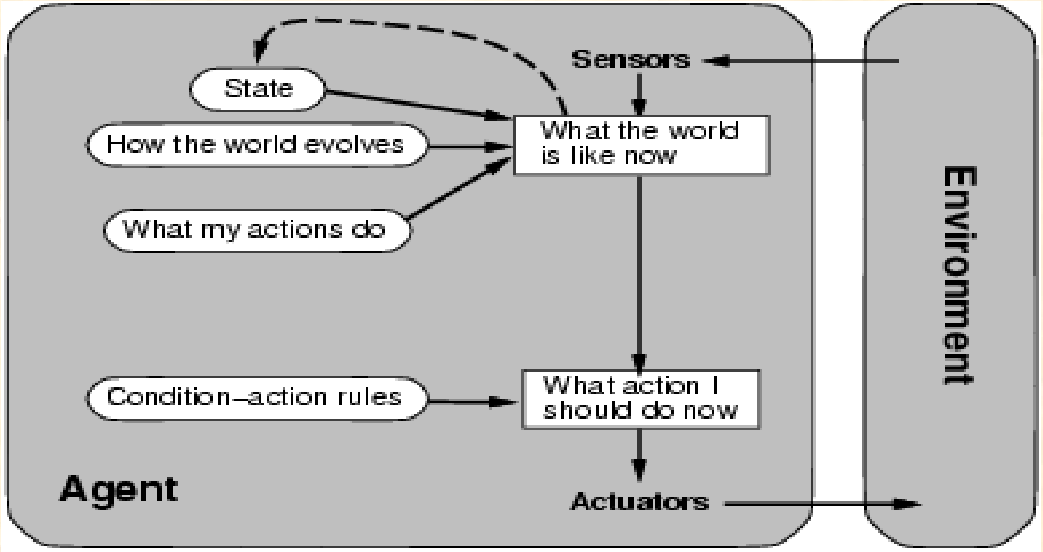
- Select action on current percept
- Ignore percept history
- Actions based on condition-action rule
- Problems?? – works only in fully observable environment

Simple reflex agents



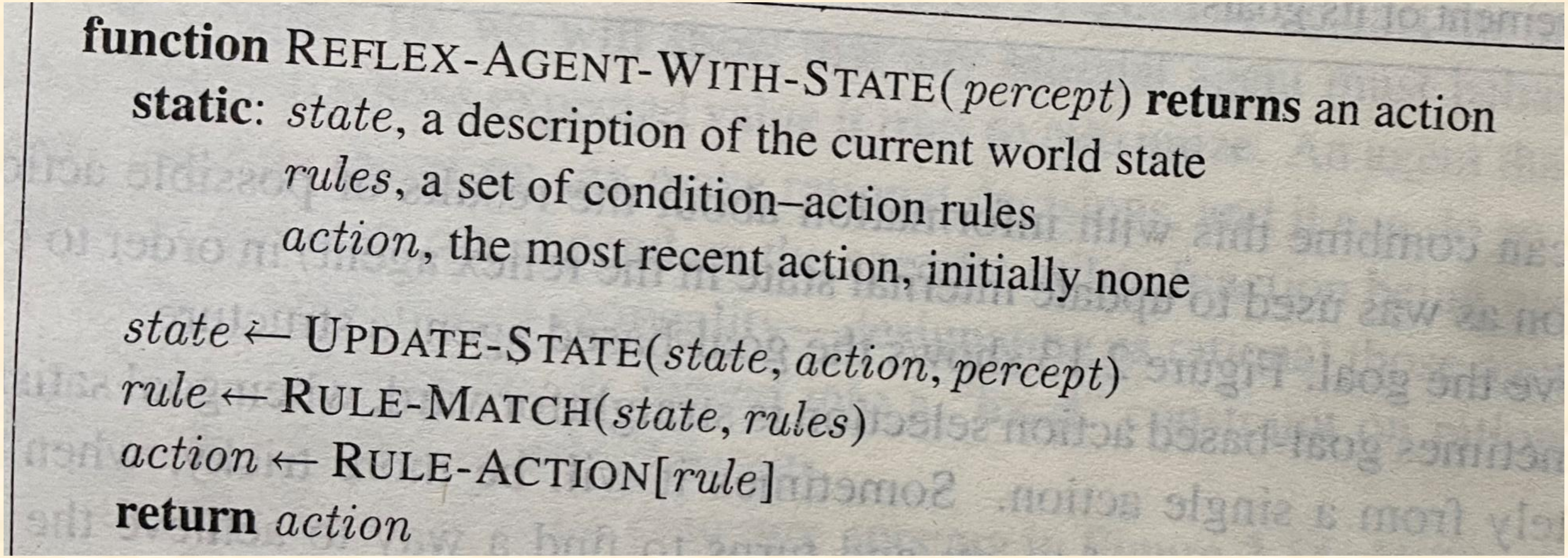
```
function SIMPLE-REFLEX-AGENT(percept) returns an action
    static: rules, a set of condition–action rules

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

Model-based reflex agents



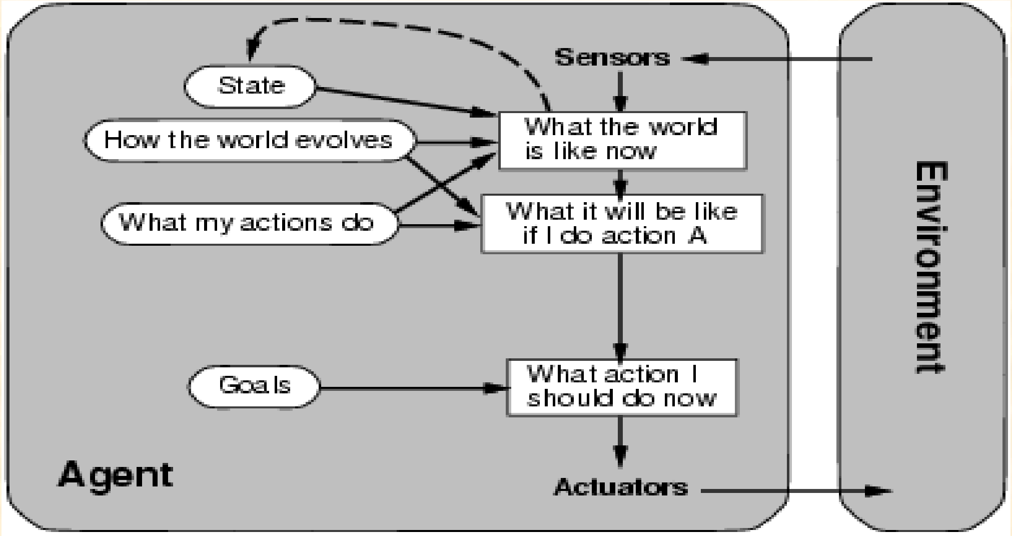
- Keep track of current state of the world agent can't see
- Maintain an internal state that depends on percept history n reflect unobserved aspects
- updating internal model requires knowledge about how the world evolves independent of the agent and how agent actions affect the world
- Model of the world → knowledge about how the world , hence the name

Model-based reflex agents



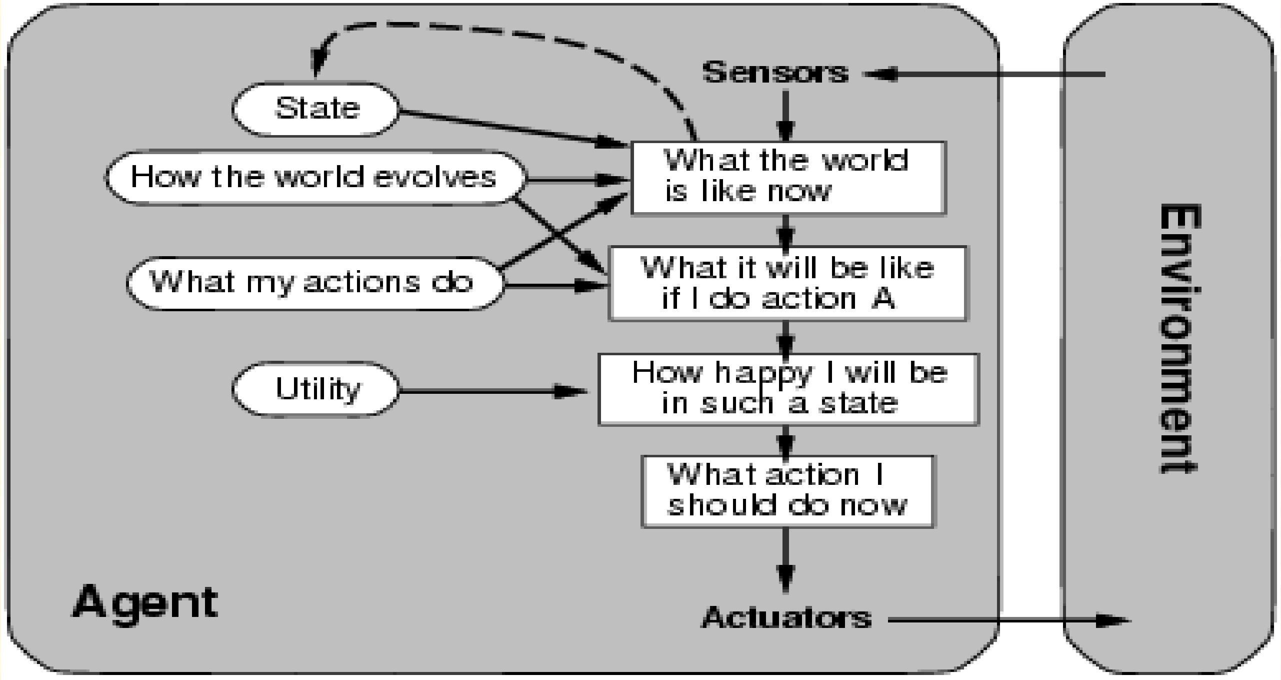
```
function REFLEX-AGENT-WITH-STATE(percept) returns an action
    static: state, a description of the current world state
            rules, a set of condition–action rules
            action, the most recent action, initially none

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

Goal-based agents


Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

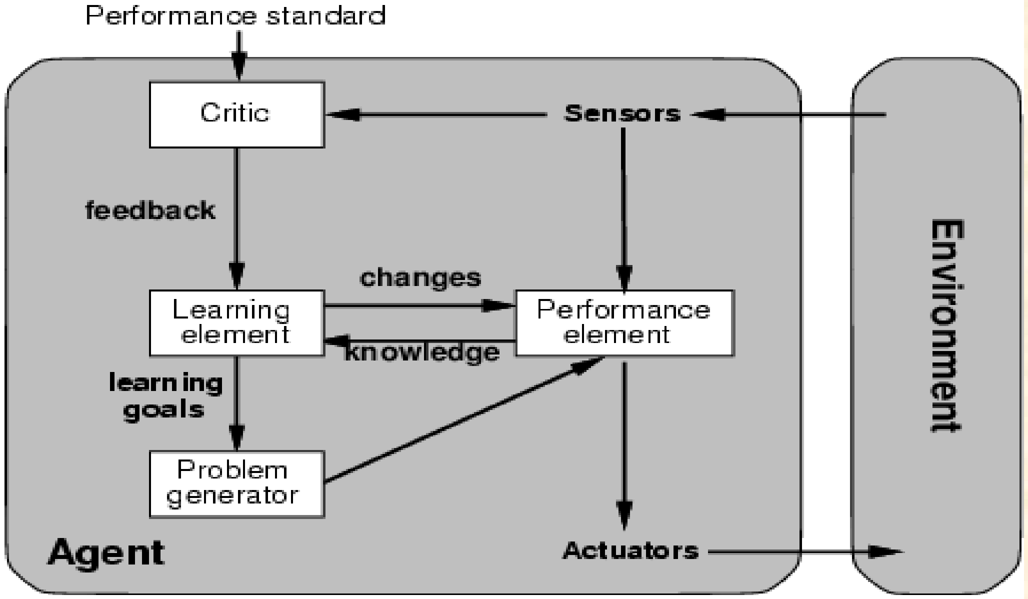
- Keep track of current state of the world and set of its goals
- Search and planning are required to achieve goals
- Decision making involves consideration of future – what will happen and will I be happy
- It is less efficient but more flexible s knowledge is explicit and can be modified
- Agents behaviour can be changed to a different location

Utility-based agents


Sensors
State
How the world evolves
What my actions do
Utility
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

- Goals alone are insufficient to generate high quality behaviour
- Goals provide crude distinction between happy and unhappy state – need how happy
- Instead higher Utility of agent is preferred
- utility function maps a state to real number – degree of happiness
- Allows rational decision where goal is inadequate : in conflicting goals and in selecting goals that can be achieved

Learning agents



- Learning allows agent to operate in initially unknown environment
- Learning element : making improvements
- Performance element : selecting external actions
- Critic : provides feedback on how the agent is doing and determines how the performance element should be modified
- Problem generator: suggesting actions

Conclusion

- Agents can have variety of components
- Each component can be presented in a variety of ways
- A variety of learning methods but one unifying theme
- Learning in intelligent agent is a process of modification of each component to bring them into a closer agreement with available feedback information thereby improving overall agent performance