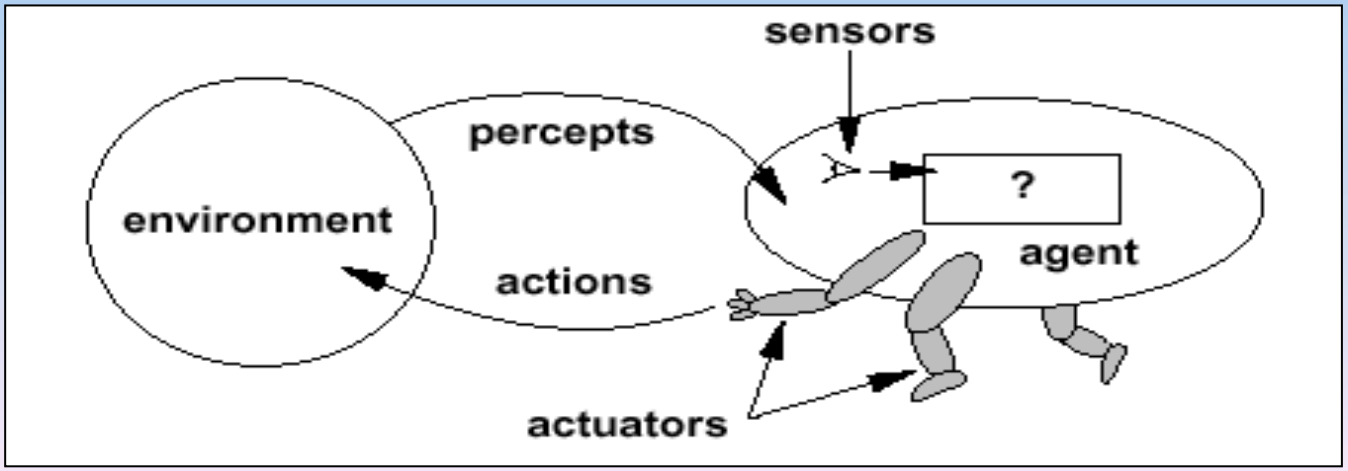
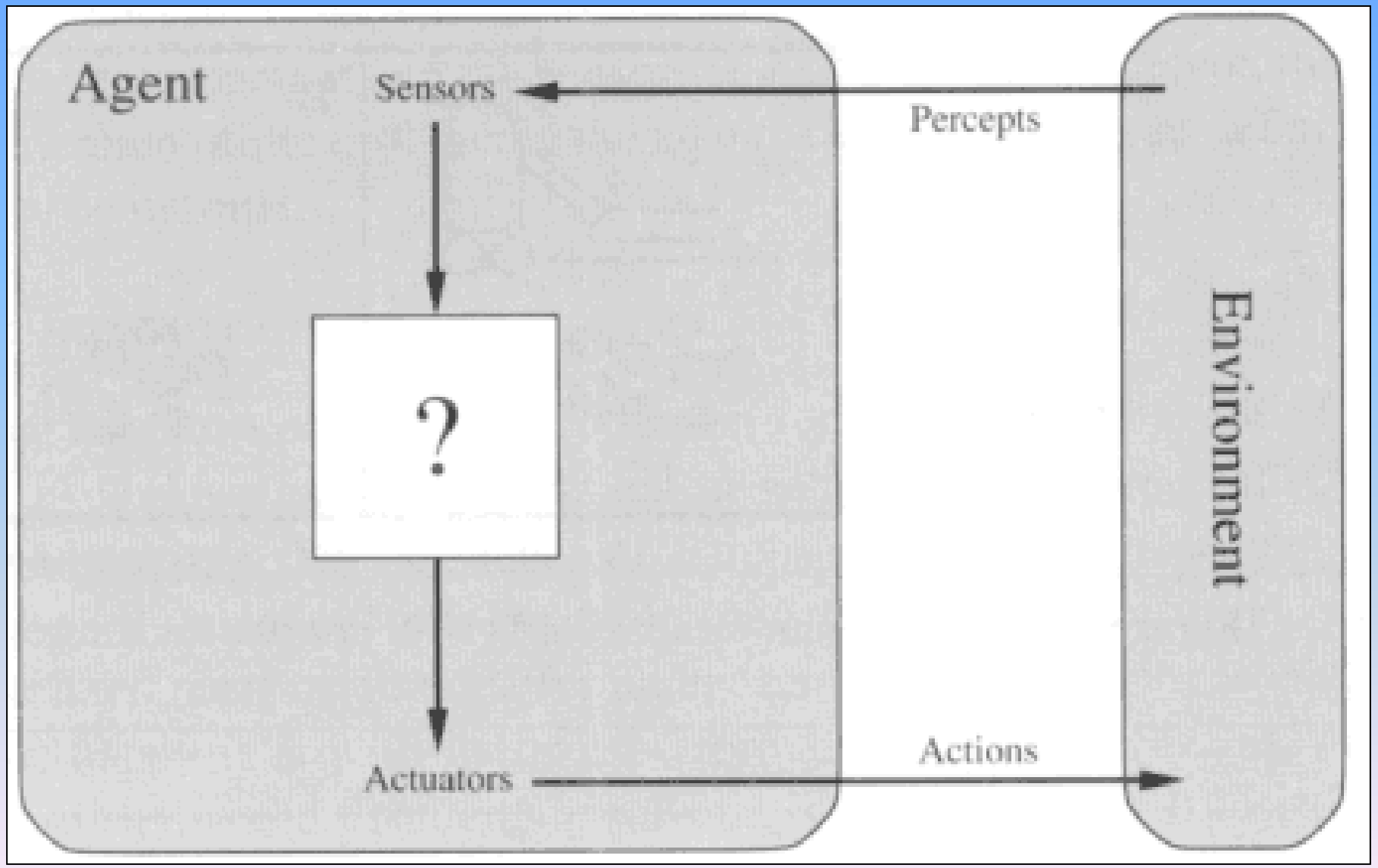
Intelligent Agents

Intelligent Agent

- An **agent** is something that **perceives** it's environment through **sensors** & **acts** upon it through **actuators.**
  - A **robot** with cameras (sensors) & robotic arm (actuators)
  - An **AC** detecting (sensors) & regulating (actuators) room temperature
  - An **automatic Vacuum Cleaner** detecting (sensors) dirty & cleaning (actuators) dirty.



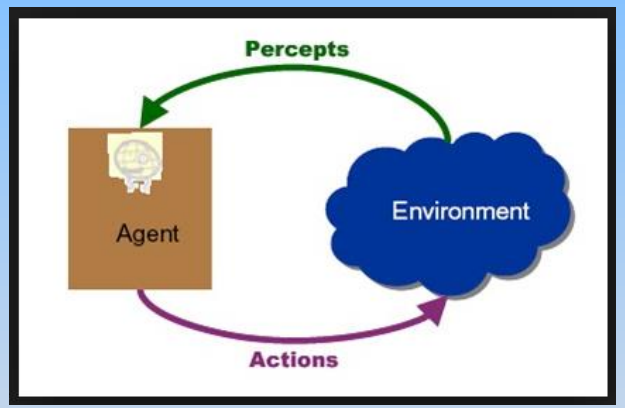
Intelligent Agent - Diagram



Some simple terms

- **Percept**

  Agent's **perceptual inputs from environment** at any given instant



- **Percept sequence**

  Complete **history** of everything that the agent has ever perceived *(**Ex**: temp or location reading at short time intervals…)*

Agent function & program

- Agent's behavior is abstract *mathematically* described by **Agent function**

  A **function** mapping any given percept sequence to an action

Percept Seq.	Action

Agent →

Architecture
(Data)

Program
(Algorithm)

- ***Agent Function***
- ***Agent Program***

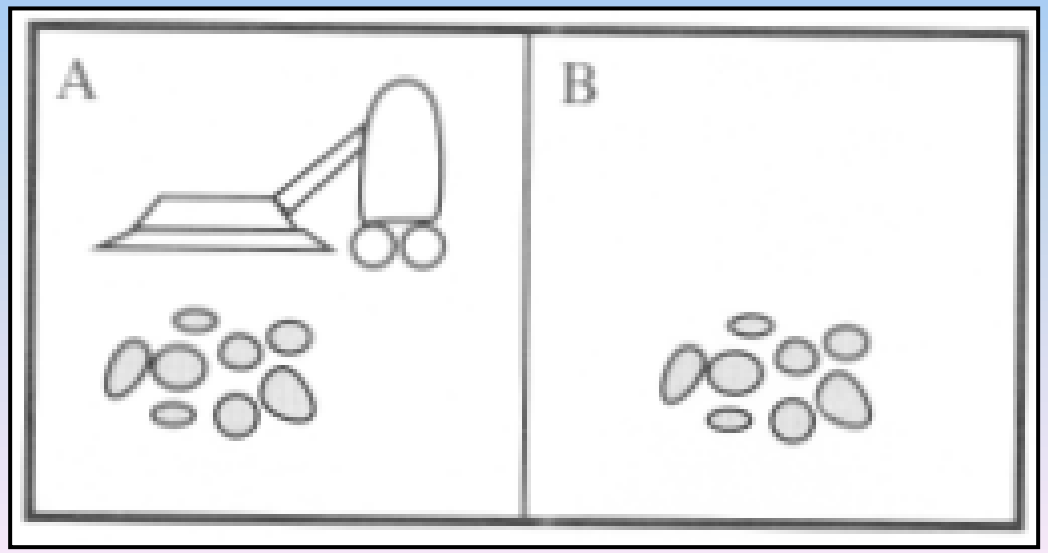
- Agent's behavior is *Practically* described by

  Agent program

  The real implementation

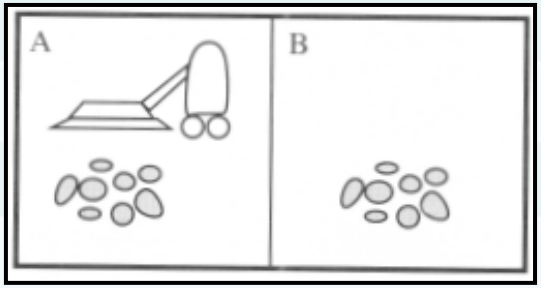
Example: Vacuum-World

- **Perception**:
  - *Clean or Dirty?*
  - *Where it is in? (location **A** & **B**)*
- **Actions**: *Move left, Move right, Suck, Do nothing*



Vacuum-World

- **Agent function** mapping percept sequences to actions partial tabulation

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
- - -	

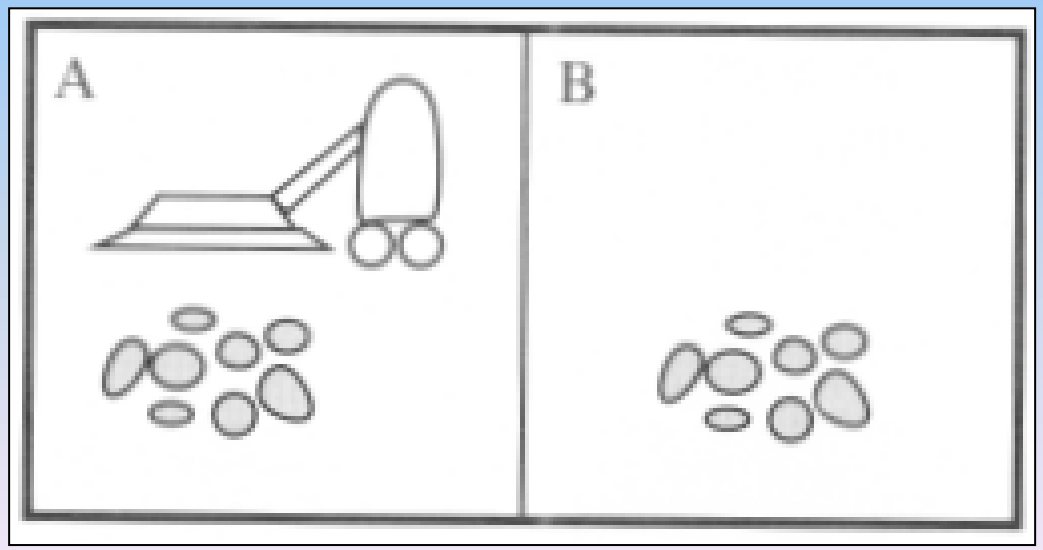
Agent Program - implementation

Function Reflex-Vacuum-Agent ([*location, state*]) return an action

If *state = Dirty* **then return *Suck***

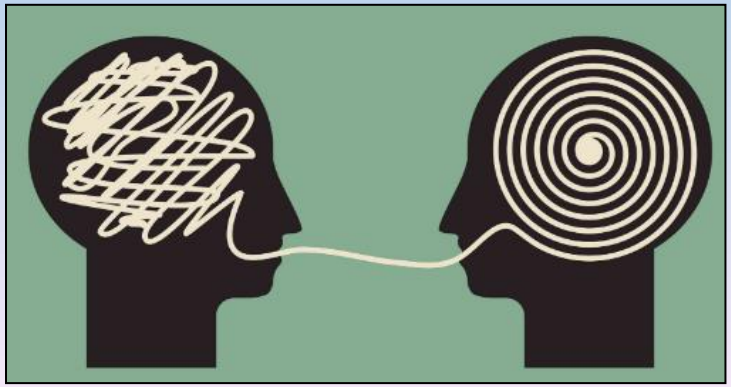
else if *location = A* **then return *Right // state = clean***

else if *location = B* **then return *Left***



Concept of Rationality

- **Rational agent**
  - One that does the right thing
  - Every entry in the agent function table is **correct (rational)**
- **What is correct?**
  - The actions that cause the agent to be most successful
  - Need ways to measure success



Concept of Rationality

- How is interpreted as “good behavior of an agent”,
  - Performance Measure
  - Rationality
  - Omniscience
  - Learning
  - Autonomy

Performance measure

- **Performance measure**
  - An ***Objective function*** determines:

    Criteria of Success for an agent

    Ex: Passed the exam if scored 45%

    Success if 90% clean

- An agent, based on its percepts performs **action sequence** if result is desirable, it is said to be performing well

- No "universal performance measure" exists for all agents

- A general rule:
  - Design "**performance measures**" according to
    - "**What** is required to be achieved" in the environment.
    - Rather than "**How** the agent should behave"

- **Ex**. in vacuum-cleaner
  - We want the "**floor clean**"
  - We don't restrict "**how the agent behaves**"

Rationality

- **What is rational** behavior at any given time depends on **Four factors:**

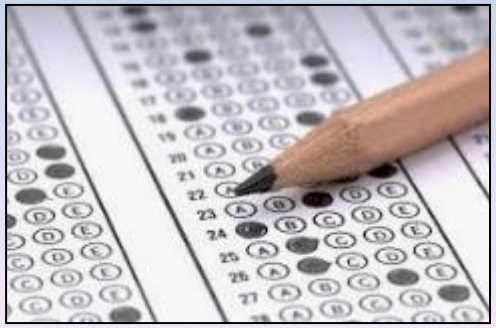
  1. The "**performance measure**" defining the "criterion of success"
  2. The agent's "**prior knowledge**" of the environment
  3. The "**actions**" that the agent can perform
  4. The agent's "**percept sequence**" up to now

- For each possible "percept sequence":

  A **rational agent** should select an action expected to "**maximize performance**" with whatever "**built-in knowledge**" the agent has

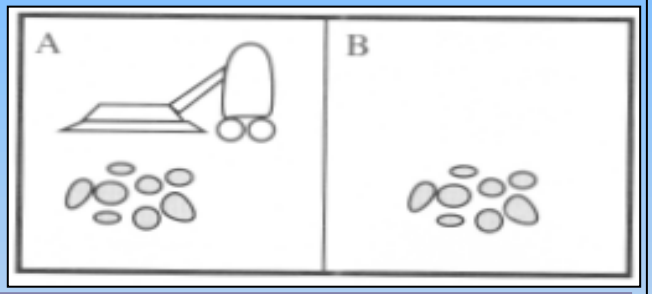
---

- **Ex**. An exam
  - *Maximize marks, based on the given questions & own knowledge*



Example rational agent – Vacuum Cleaner

- **Performance measure**
  - Awards "one point" for each "clean square" at each time step, measured over 10000 time steps
- **Prior knowledge about the environment**
  - The "geography" of the environment
    - Only two squares
  - The "*effect*" of the actions



- **Actions it can perform**
  - **Left, Right, Suck & No Operation**
- **Percept sequences**
  - "Where" is the agent? (A or B)
  - Whether the "location contains dirt"? (Dirty or Clean)
- *Under this circumstance, the agent is rational*

Omniscience Agent

- Knows the ***actual* outcome** of it's actions in advance
- This is impossible in real world
- **Ex** – *While crossing an empty street, man dies of the fallen cargo door from an aircraft → was the man irrational?*
- *Outcome depends only on current percept not past percept sequence*
- Rational agents are not *omniscient*

Learning Agent

• If an **agent** depends on "current percept" as well as "past percept sequence". This is called **learning**

• After "*experiencing an episode*", the agent should **adjust** it's behaviors to "*perform better*" next time

Autonomous Agent

- If an agent just relies on the "prior knowledge of it's designer" rather than its "own percepts" then the agent lacks ***autonomy***
- An autonomous agent should learn to compensate for partial prior knowledge

Software Agents

- In some cases the environment is not real world but "artificial"
  - ***Ex****: flight simulator, video games, Internet*
- Those agents working in these environments are called "Software agents" (**softbots**) - *All parts of the agent are software*

Task environment (PEAS Description)

- Task environment describes the problem. In designing an agent, the first step is to describe the task environment.
- The task environments is the problems for which the rational agent is the solution.

Specifying PEAS, describe the task environment.

P: **P**erformance (the measure success)

E: **E**nvironment (the external world in which the agent operates)

A: **A**ctuators (Implements agents action on the env)

S: **S**ensors (Reads or takes inputs from the env)

Ex-Automated taxi driver

PEAS description of Automated Taxi driver

- **Performance** (the measure success)
  How to **judge** the performance of a **automated driver**?
  Factors - Reaching correct destination, Minimizing fuel consumption, Trip time, Non-violations of traffic laws, Maximizing the safety etc.

- **Environment** (the external world in which the agent operates)
  A taxi must deal with **Road condition**, **Traffic lights**, other **Vehicles**, **Pedestrians**, **Animals**, **Road works** etc. & also interact with the **customer**

- **Actuators** (Implements agents action on the env)
  Provides control over the **steering**, **gear**, **brake** & **communicates** with customer

- **Sensors** (Reads or takes inputs from the env) Detect other **vehicles**, **road situation (camera)**, **GPS** to know the location of the taxi

Task environments - Automated taxi driver

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, Fast, Legal, Comfortable trip, Maximize profit	Roads, Other Traffic, Pedestrians, Customers	Steering, Accelerators, Brake, Signal, Horn, Display	Cameras, Sonar, Speedometer, Accelerometer, Engine Sensors, Keyboard

PEAS Description of the task environment for an Automated Taxi

Agent type	P	E	A	S
Medical Diagnosis System	Healthy patient, minimize costs, lawsuits	Patient, Hospital Staff	Display questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's Answers.
Interactive English Tutor	Maximize student's score on test.	Set of students, testing agency.	Display exercises, suggestions, corrections	Typed words.
Satellite Image Analysis System	Correct categorization	Downlink from orbiting satellite	Display categorization of scene.	Colour pixel arrays.
Refinery Controller	Maximize purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Taxi Driver	Safe: fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display.	Cameras, sonar, speedometer, GPS, odometer, accelerometer,

Properties of Task Environment

- Fully Observable Vs. Partially Observable
- Single Agent Vs. Multiagent
- Deterministic Vs. Non-deterministic
- Episodic Vs. Sequential
- Static Vs. Dynamic
- Discrete Vs. Continuous
- Known Vs. Unknown

Structure of agents

Agent

Architecture
(Components)

Program
(Algorithm)

- ***Agent Function***
- ***Agent Program***

Structure of agents

- **Agent = architecture + program**
  - **Architecture** = the components of agent *(sensors, actuators..)*
  - **(Agent) Program** = the functions (job of AI) that implement the agent actions based on the current percept

Agent programs

- **Input for "Agent Program"** - only the current percept
- **Input for "Agent Function"** - the entire percept sequence
- Can be implemented as a **look up table**

function TABLE-DRIVEN-AGENT(*percept*) **returns** *action*

static: *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*

Vacuum-World

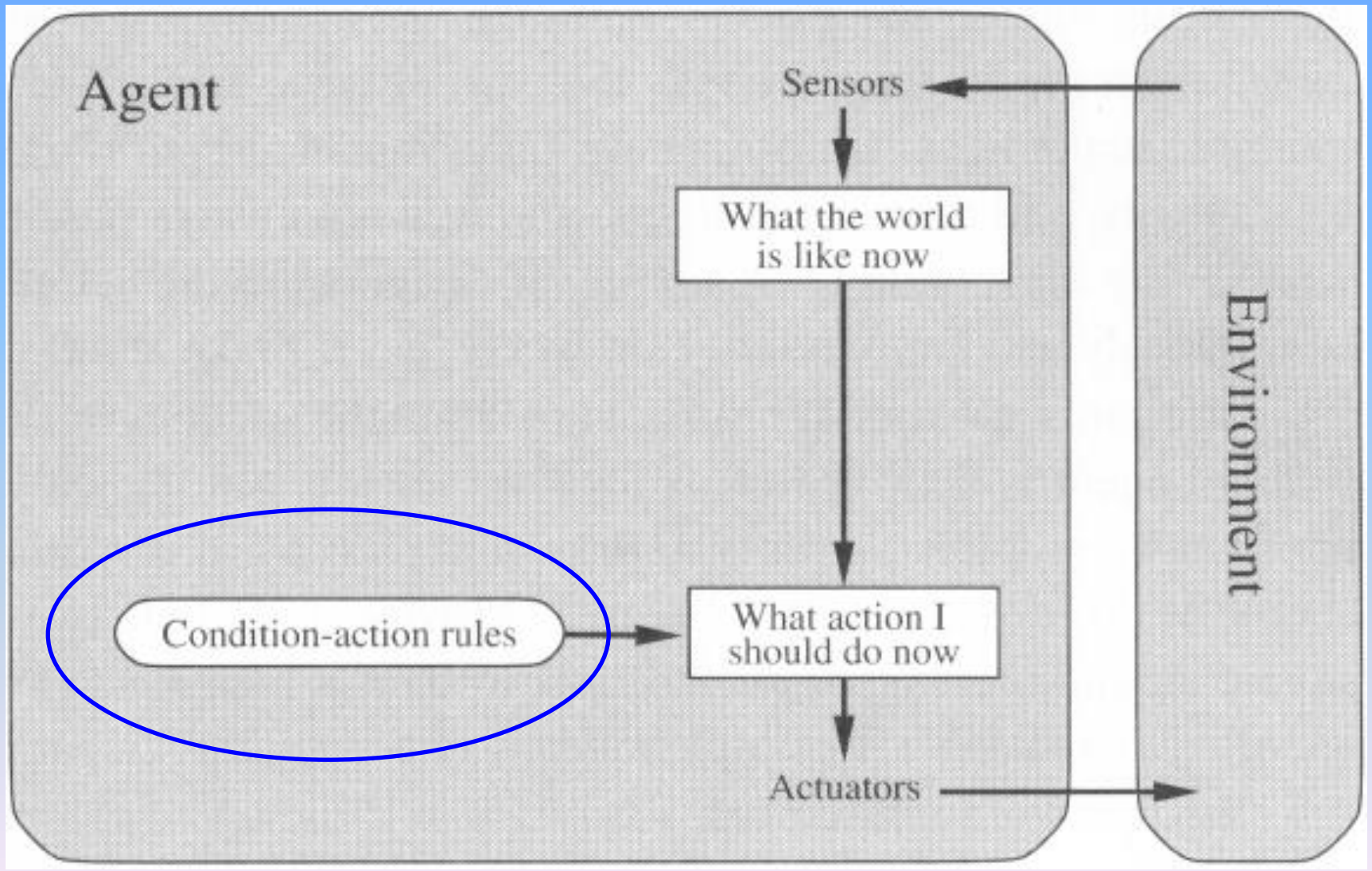
- **Agent function** mapping percept sequences to actions partial tabulation

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
- - -	

Types of Agent programs

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

1. Simple reflex agents



1. Simple reflex agents (for fully observable env)

- It uses just ***condition-action rules***
  - The rules are in form of "if … then …"
  - Efficient but have narrow range of applicability
  - Works only if the environment is "fully observable"

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

  state ← INTERPRET-INPUT(*percept*) // From percept get state
  rule ← RULE-MATCH(*state*, *rules*) // From State get rule
  action ← RULE-ACTION[*rule*] // From rule find action
  return *action*

Example: A Simple Reflex Agent in Nature

percepts
(size, motion)

RULES:

(1) If small moving object,
then activate SNAP

(2) If large moving object,
then activate AVOID and inhibit SNAP

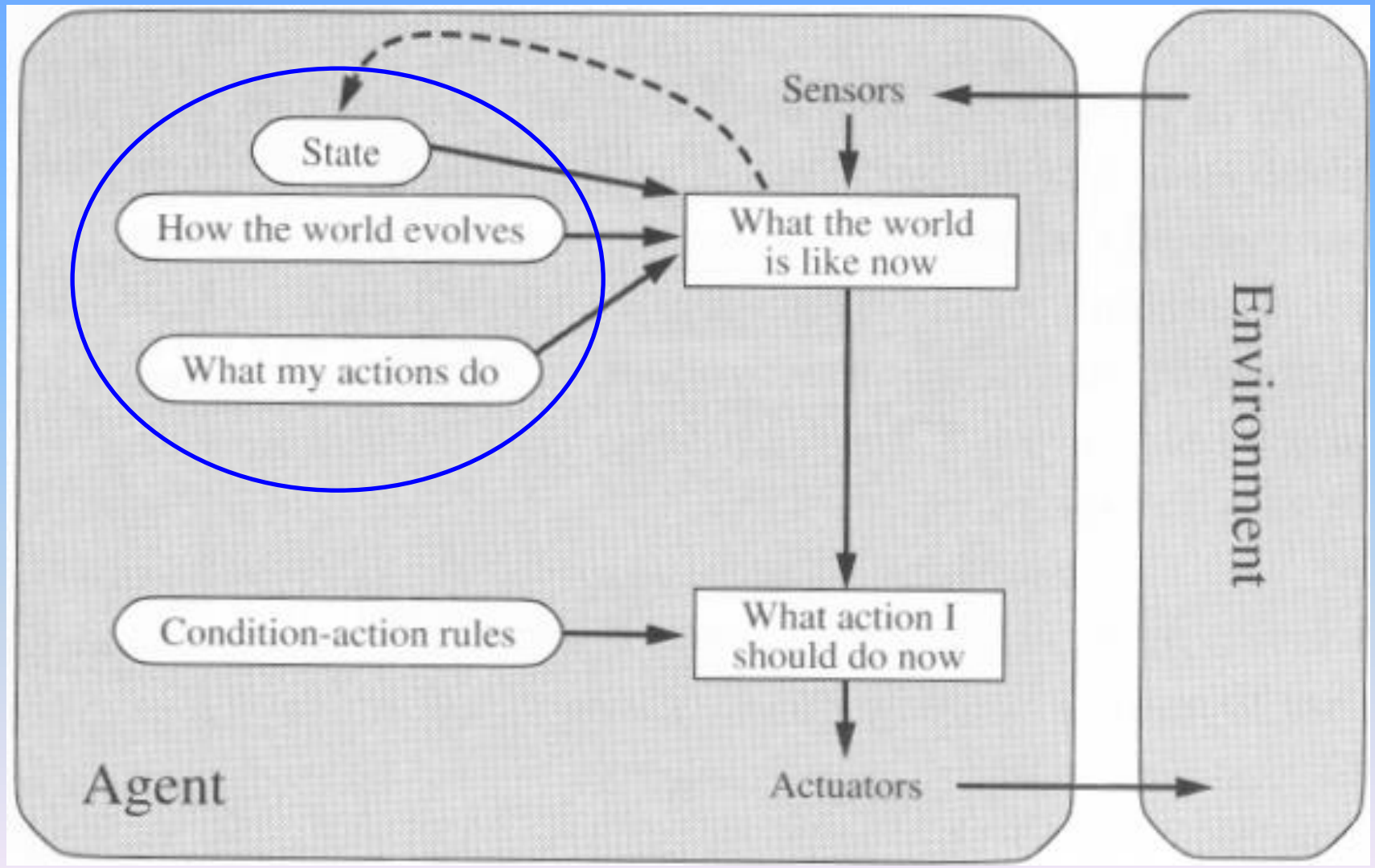
ELSE (not moving) then NOOP

needed for completeness

Action: SNAP or AVOID or NOOP

2. Model-based Reflex Agents

(for partially observable env)



2. Model-based Reflex Agents (for partially observable env)

- **For the world that is "partially observable"**
  - the agent has to keep track of it's own state
- **We also require two more types of knowledge**
  - How the *world evolves independent of the agent*
  - How the agent's *actions affect the world*

Example – Self-driven car

Example Mapping Table

States car parked, Started, running, halted…

IF	THEN
Saw an object ahead, turned right, and it's now **road clear** ahead	**Go straight**
Saw an object ahead, turned right, and **another object ahead**	**Slow down**
See **no objects** ahead	**Go straight**
See **dead end**	**Take U Turn**

Model-based Reflex Agents

```
function REFLEX-AGENT-WITH-STATE(percept) returns action
  static: state, a description of the current world state
          rules, a set of condition-action rules

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

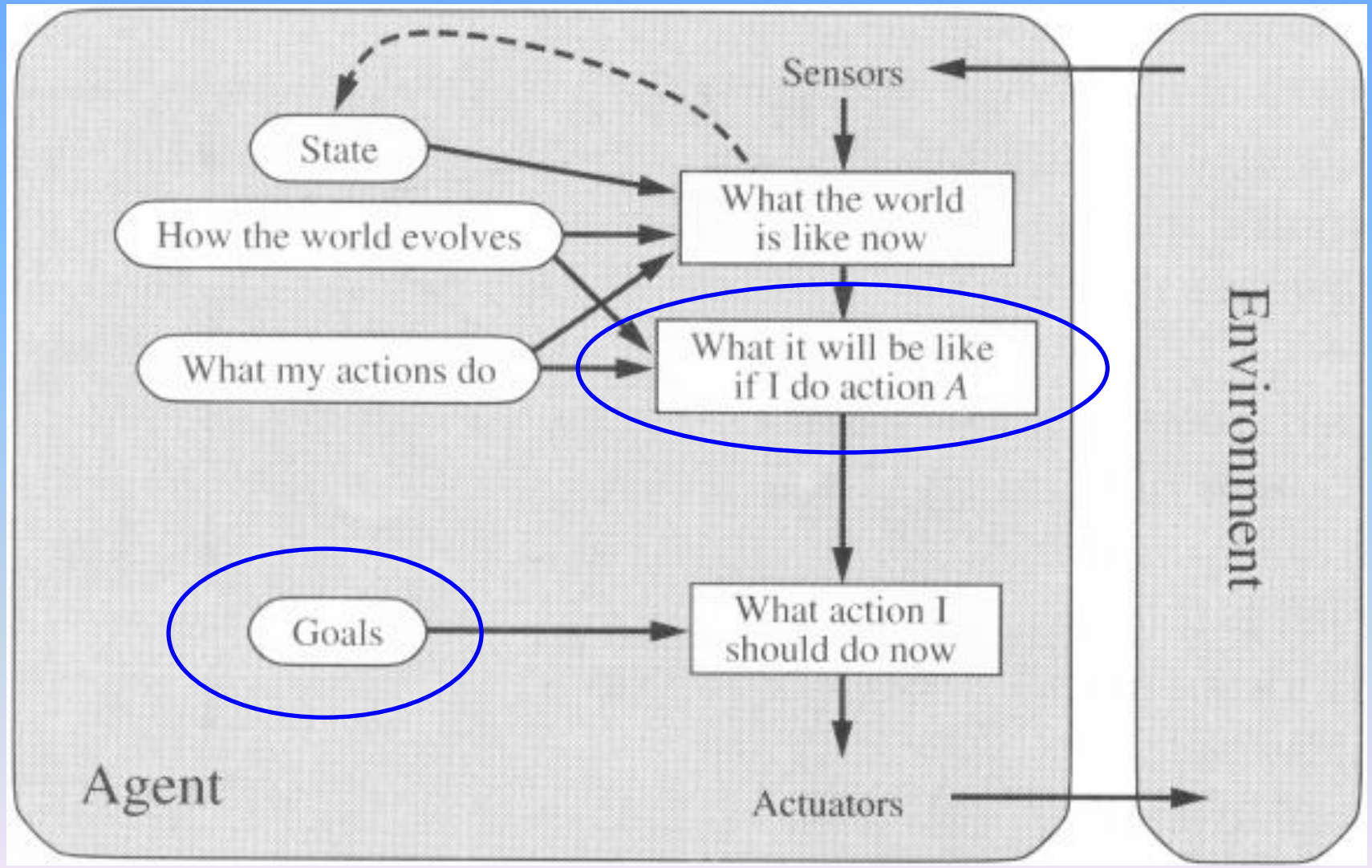
// From **current state** & **percept** get **next**

// From **State** get **rule**.

// From **rule** find the **action**.

// From **new state** & **action** update **next s**

3. Goal-based agents

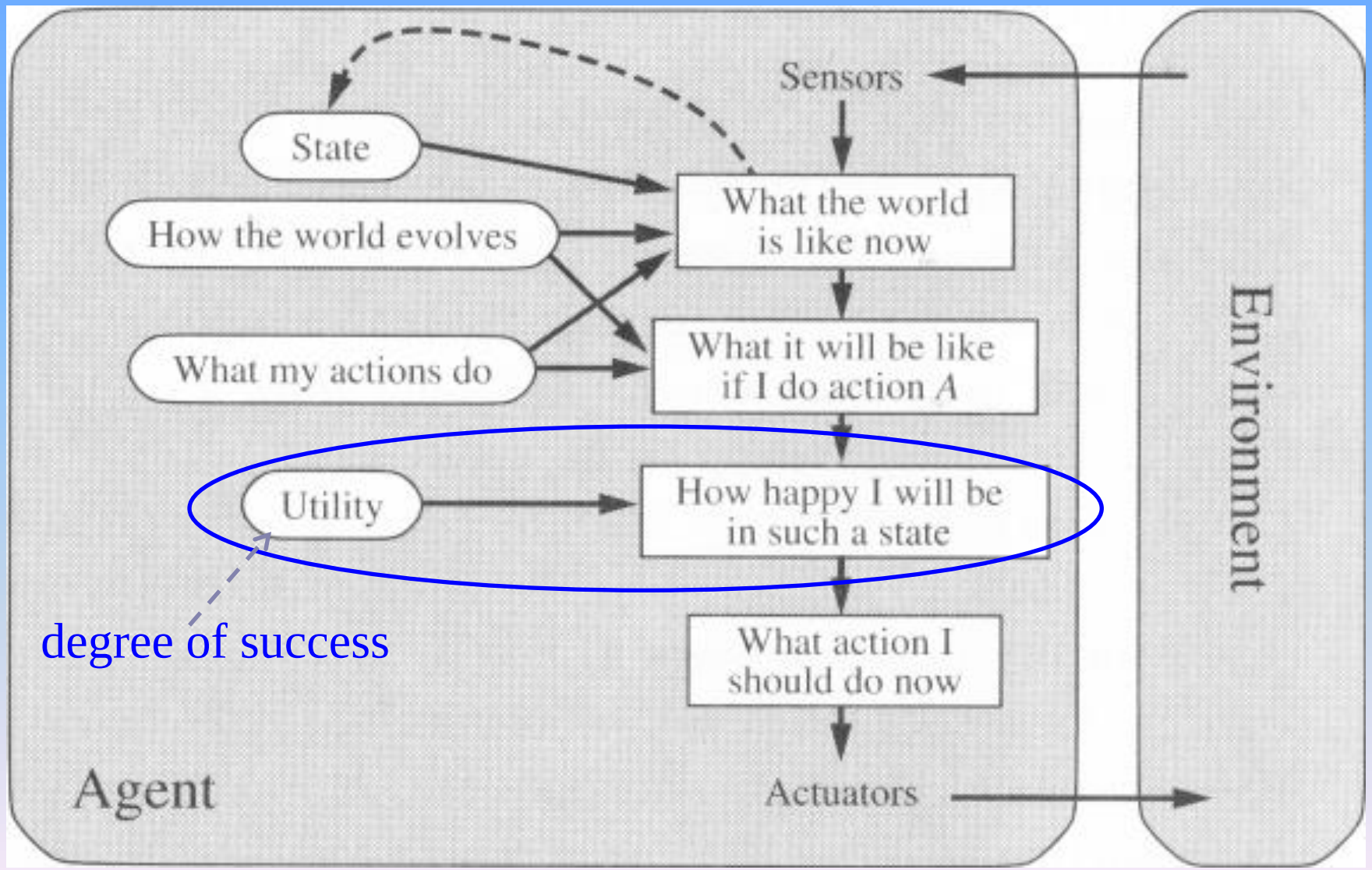


3. Goal-based agents

- The **objective** is to achieve the **goal**
- Current state of the agent & action are not enough to reach the goal
- Need to find out the action sequences required to achieve the goal
- **Choose Actions** that will lead to the **goal**, based on
  - the current state
  - the current percept

- **Conclusion**
  - Goal-based agents are less efficient but more flexible
  - Searching & Planning (Two other sub-fields in AI)

4. Utility-based agents

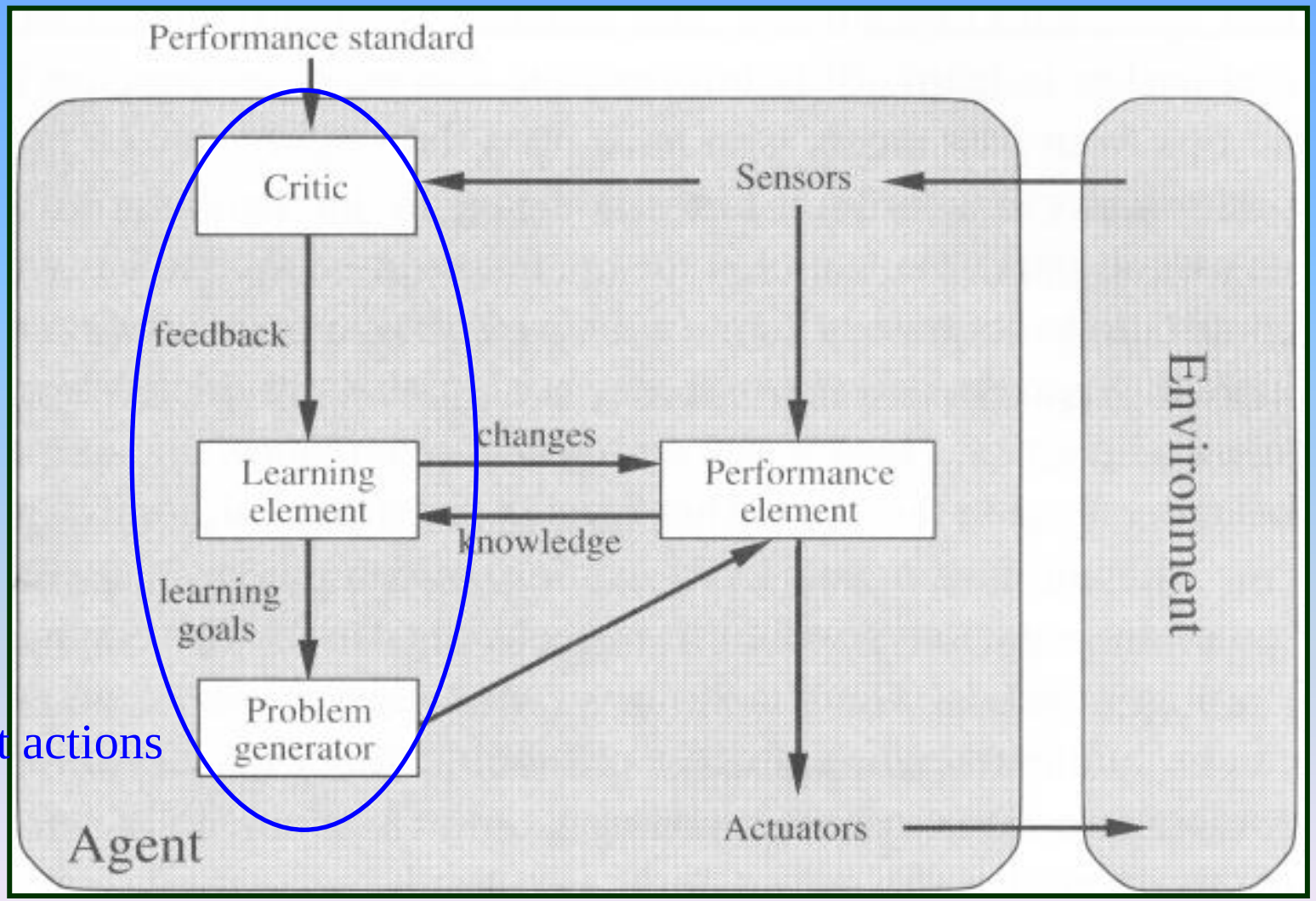


4. Utility-based agents

A2

- Reaching the **Goals** is not enough. Achieving the goal in **a better way** is important
- A **no.** of **action sequences** are required to **achieve the goal**
  - If **goal** => **success**, then **utility** => **degree of success**
- If **State A** has higher utility then it is preferred more than others
- When there are several goals & none of them can be achieved with certainty, **Utility** provides a way for the decision-making

5. Learning Agents



Suggest actions

Learning Agents

- After an agent is **programmed**, it can not work immediately
  - It **needs to be taught**
  - **Teach** it by giving it **a set of examples (Training set)**
  - **Test** it by using **another set of examples (Test set)**
  - We then say the agent - A **learning agent**

- **Four conceptual components**
  - **Learning element** – For **Making continuous improvements**
  - **Performance element** - Selecting **proper actions**
  - **Critic** - Tell **how well the agent is doing**
  - **Problem generator - Suggests actions** that will lead to new & informative experiences


The End