Intelligent Agent

Course Code: AFI 124

Intelligent Agent

To Build: Goal is to build an agents—systems that can reasonably be called intelligent

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

Behaviour: How well an agent can behave depends on the nature of the environment; some environments are more difficult than others.

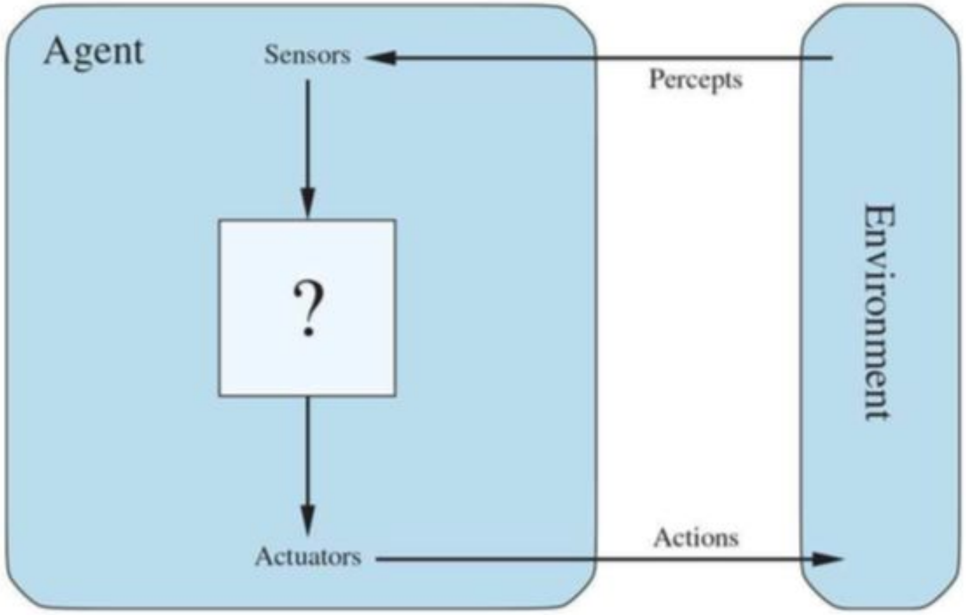
Agents and Environments

Human agent: A human agent has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators.

Robotic agent: A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators.

Software agent: A software agent receives file contents, network packets, and human input (keyboard/mouse/touchscreen/voice) as sensory inputs and acts on the environment by writing files, sending network packets, and displaying information or generating sounds.

Environment: The environment could be everything—the entire universe! In practice it is just that part of the universe whose state we care about when designing this agent—the part that affects what the agent perceives and that is affected by the agent's actions.



Agents interact with environments through sensors and actuators.

Agents and Environments

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

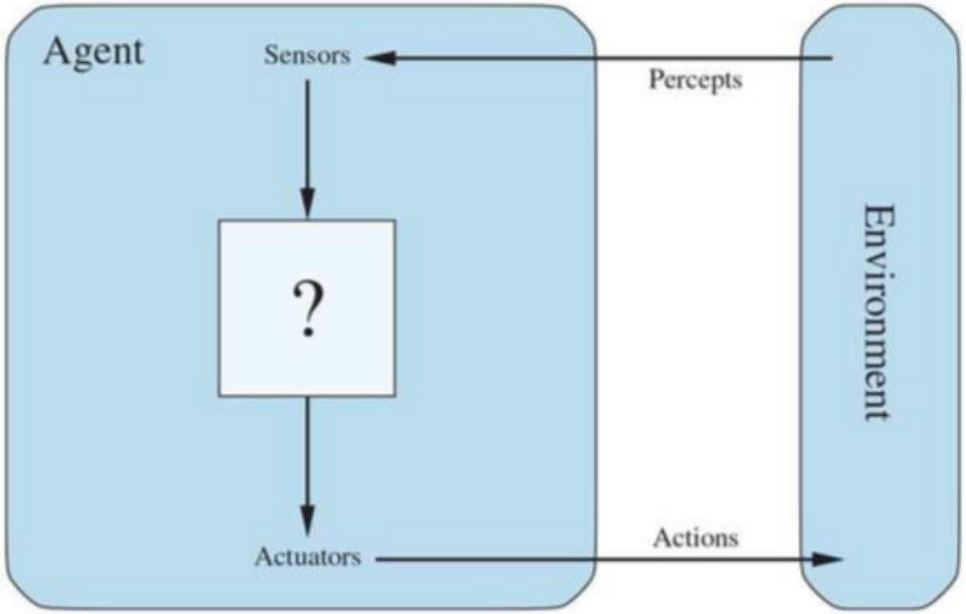
Percept: Percept refers to the content an agent's sensors are perceiving.

Actions: an agent's choice of action at any given instant can depend on its built-in knowledge and on the entire percept sequence observed to date, but not on anything it hasn't perceived.

Sensors: The environmen of the system is observed by intelligent agents through sensors.

Actuators: These are components throught which energy is converted to motion. They perform the role of controlling and moving a system.

Environment: Environment is the surrounding of the agent.



Agents interact with environments through sensors and actuators.

Example: the vacum cleaner world

The vacuum-cleaner world, which consists of a robotic vacuum-cleaning agent in a world consisting of squares that can be either dirty or clean.

Figure shows a configuration with just two squares, A and B. The vacuum agent perceives which square it is in and whether there is dirt in the square.

Steps:

- The agent starts in square A.
- The available actions are to move to the right, move to the left, 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.

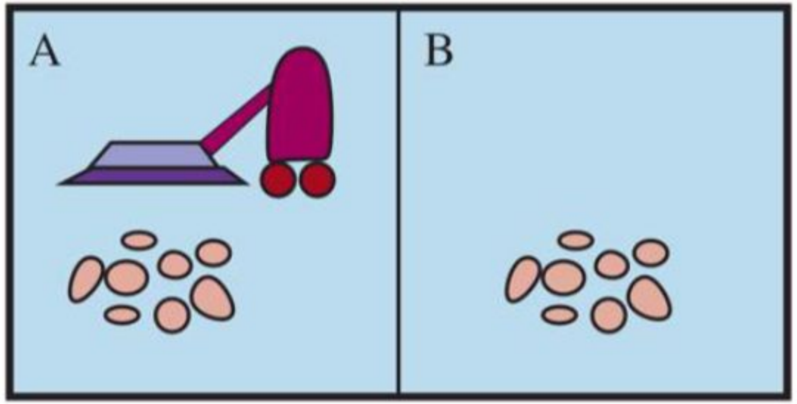
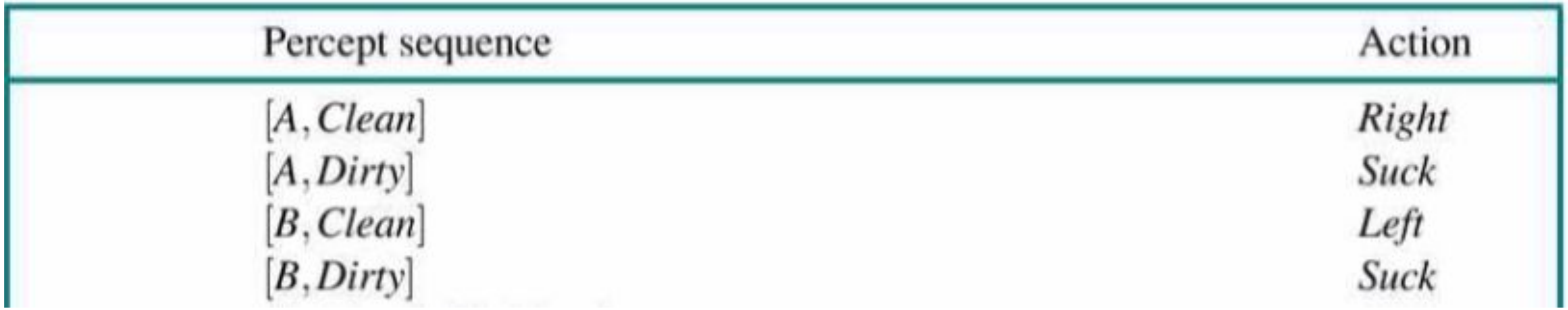


Fig: A vacuum-cleaner world with just two locations. Each location can be clean or dirty, and the agent can move left or right and can clean the square that it occupies. Different versions of the vacuum world allow for different rules about what the agent can perceive, whether its actions always succeed, and so on.

Percept and action



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

The agent cleans the current square if it is dirty, otherwise it moves to the other square. Note that the table is of unbounded size unless there is a restriction on the length of possible percept sequences.

Looking at Table, we see that various vacuum-world agents can be defined simply by filling in the right-hand column in various ways. The obvious question, then, is this: What is the right way to fill out the table? In other words, what makes an agent good or bad, intelligent or stupid?

Good Behaviour: The concept of Rationality

A **rational agent** is one that does the right thing. “*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. “*

Evaluation of Good Behaviour:

1. Performance Measures
2. Rationality
3. Omniscience, learning, and autonomy

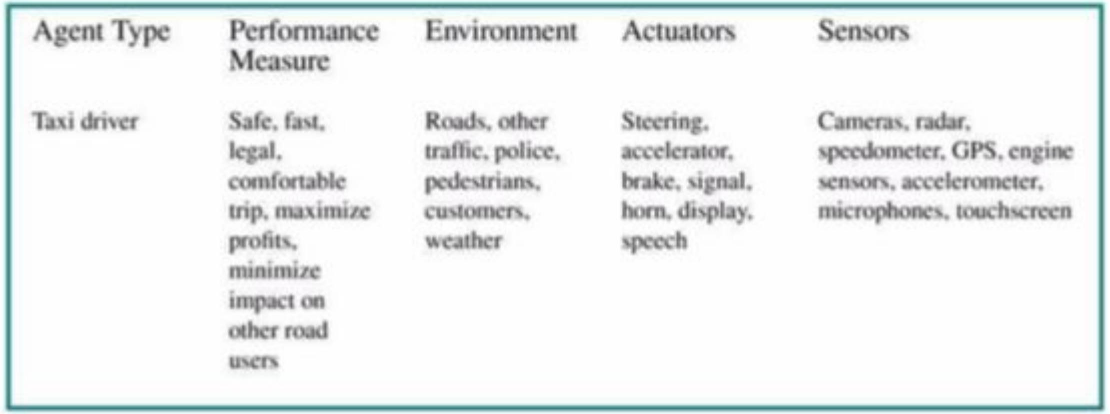
An **omniscient agent** knows the *actual* outcome of its actions and can act accordingly; but omniscience is impossible in reality.

The Nature Of Environment

PEAS (Performance, Environment, Actuators, Sensors)

PEAS Example: Automated taxi driver

- what is the performance measure to which we would like our automated driver to aspire?
- what is the driving environment that the taxi will face?
- The actuators for an automated taxi include those available to a human driver.
- The basic sensors for the taxi will include one or more video cameras so that it can see.

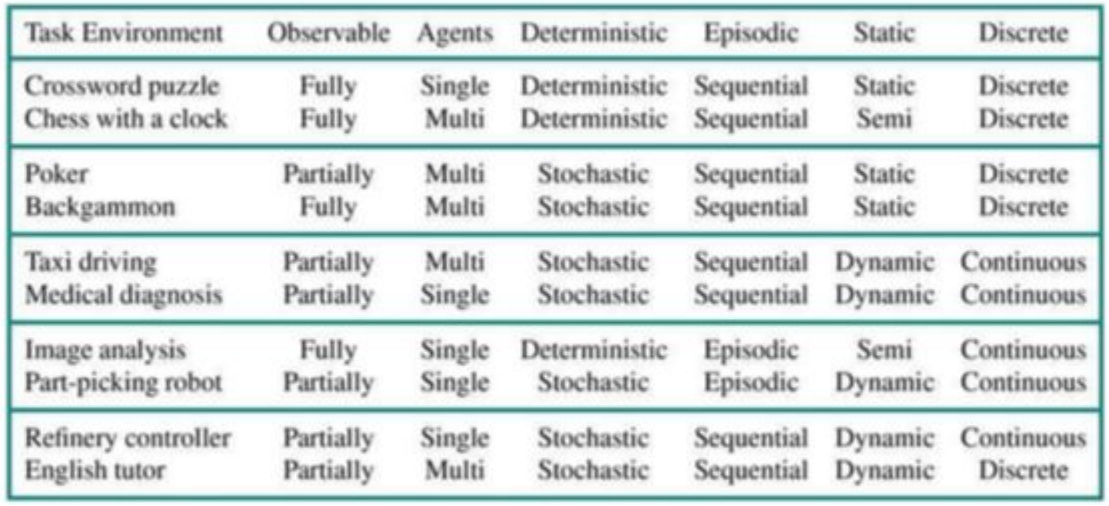


Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits, minimize impact on other road users	Roads, other traffic, police, pedestrians, customers, weather	Steering, accelerator, brake, signal, horn, display, speech	Cameras, radar, speedometer, GPS, engine sensors, accelerometer, microphones, touchscreen

PEAS description of the task environment for an automated taxi driver.

Properties/Types of environment

- FULLY OBSERVABLE VS. PARTIALLY OBSERVABLE
- SINGLE-AGENT VS. MULTIAGENT
- EPISODIC VS. SEQUENTIAL
- STATIC VS. DYNAMIC
- DISCRETE VS. CONTINUOUS
- KNOWN VS. UNKNOWN
- Deterministic vs Stochastic



Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Examples of task environments and their characteristics.

Structure of intelligent agent

The job of AI is to design an agent program that implements the agent function—the mapping from percepts to actions. We assume this program will run on some sort of computing device with physical sensors and actuators—we call this the agent architecture:

Agent = Architecture + Program

If the program is going to recommend actions like Walk, the architecture had better have legs

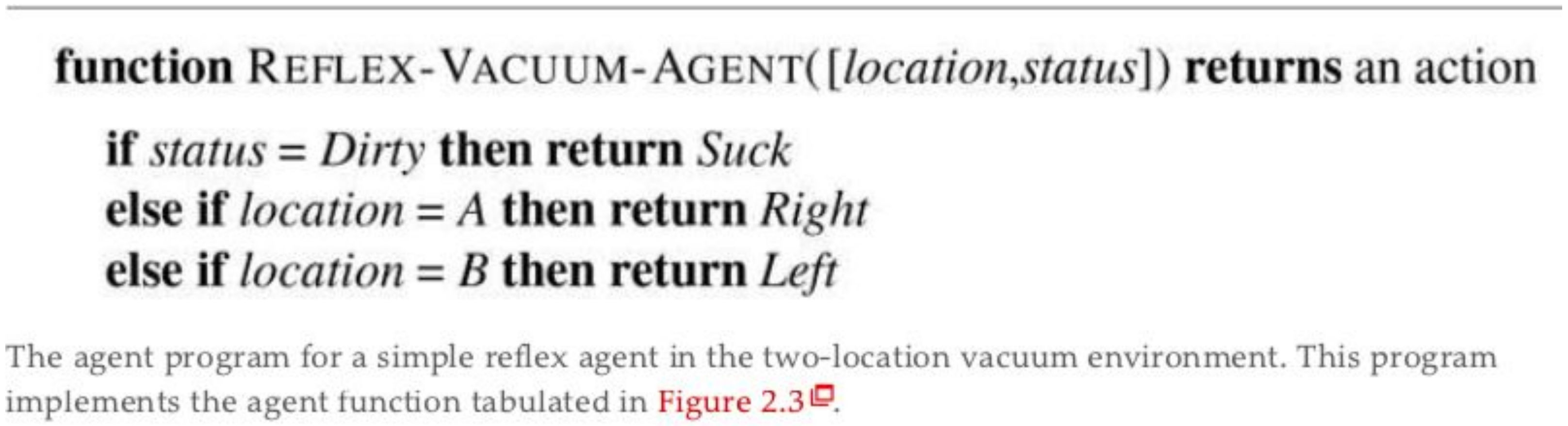
The principles underlying almost all intelligent systems

1. Simple reflex agents;
2. Model-based reflex agents;
3. Goal-based agents;
4. Utility-based agents;
5. Learning Agents

Simple reflex agents

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.

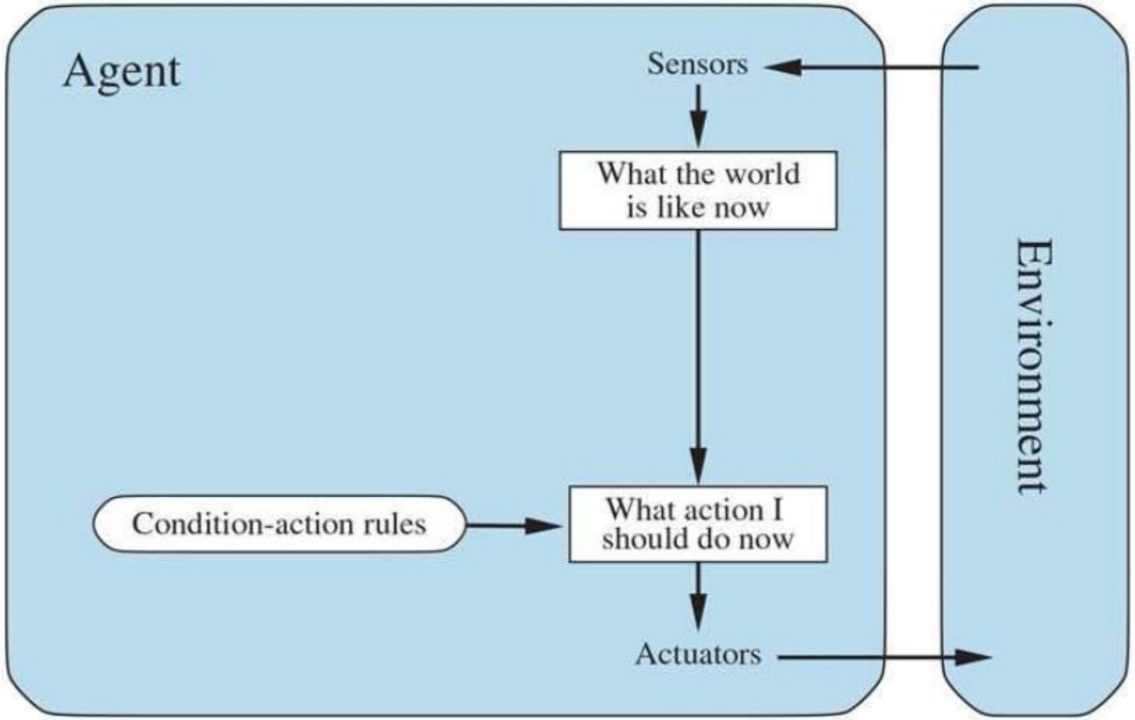
For example, the vacuum agent whose agent function is tabulated in previous table(Percept and Action) is a simple reflex agent, because its decision is based only on the current location and on whether that location contains dirt.



```
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
```

The agent program for a simple reflex agent in the two-location vacuum environment. This program implements the agent function tabulated in Figure 2.3.

Simple reflex agents Diagram



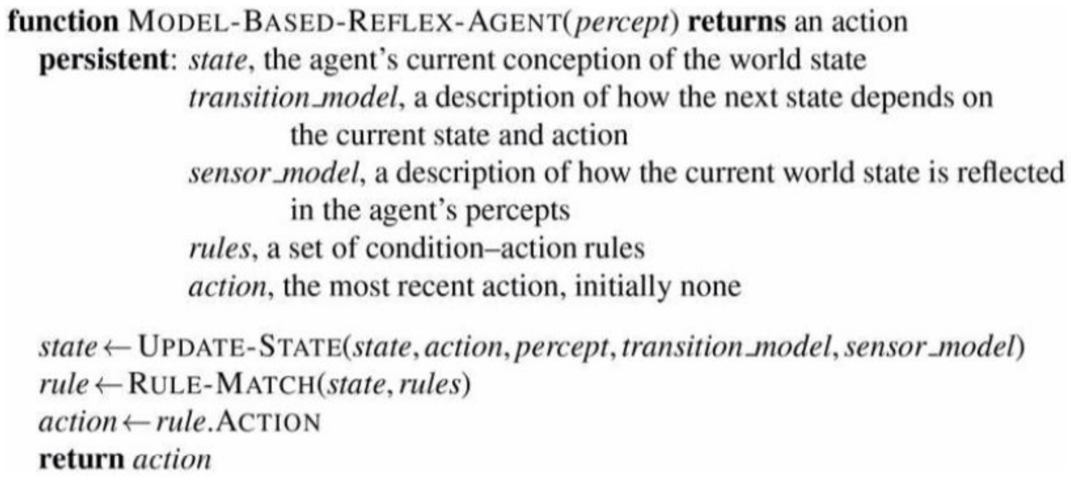
Schematic diagram of a simple reflex agent. We use rectangles to denote the current internal state of the agent's decision process, and ovals to represent the background information used in the process.

Model-based reflex agents

A **model-based reflex agent** is an intelligent agent that uses percept history and internal memory to make decisions about the "model" of the world around it.

Knowledge about “how the world works”—whether implemented in simple Boolean circuits or in complete scientific theories—is called a **transition model** of the world.

Together, the **<u>transition model and sensor model</u>** allow an agent to keep track of the state of the world—to the extent possible given the limitations of the agent’s sensors. An agent that uses such models is called a **model-based agent**.

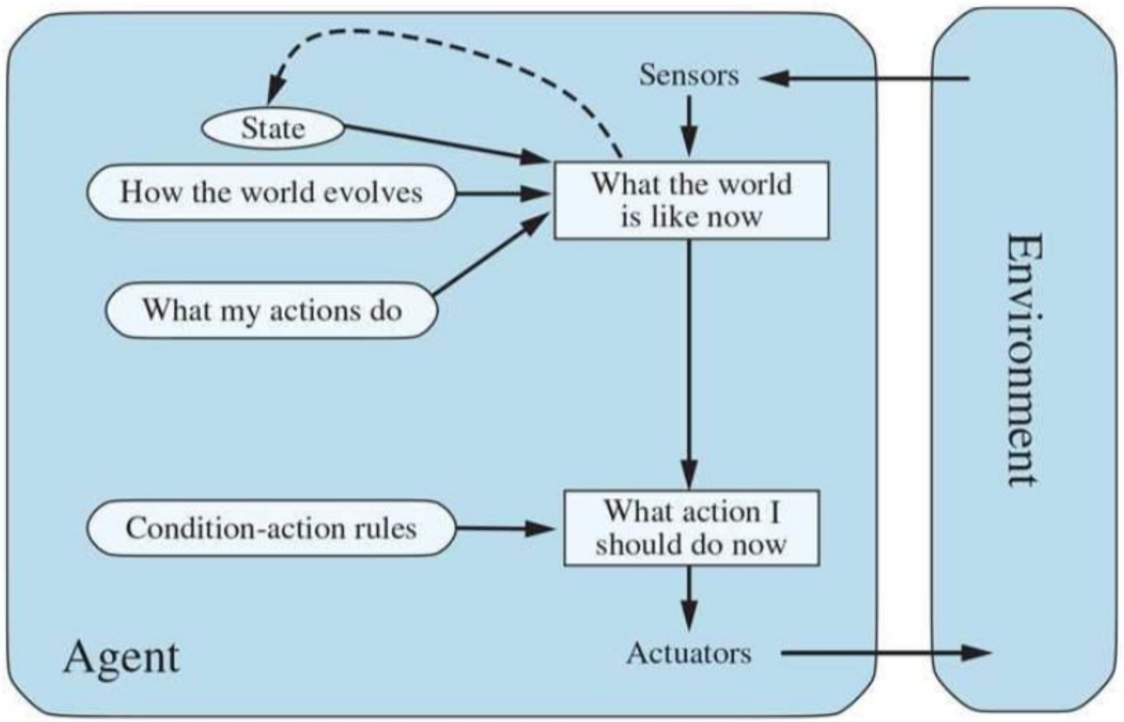


```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
   persistent: state, the agent’s current conception of the world state
               transition_model, a description of how the next state depends on
                       the current state and action
               sensor_model, a description of how the current world state is reflected
                       in the agent’s percepts
               rules, a set of condition–action rules
               action, the most recent action, initially none

   state ← UPDATE-STATE(state, action, percept, transition_model, sensor_model)
   rule ← RULE-MATCH(state, rules)
   action ← rule.ACTION
   return action
```

A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

Model-based reflex Diagram



A model-based reflex agent.

Goal-based agents

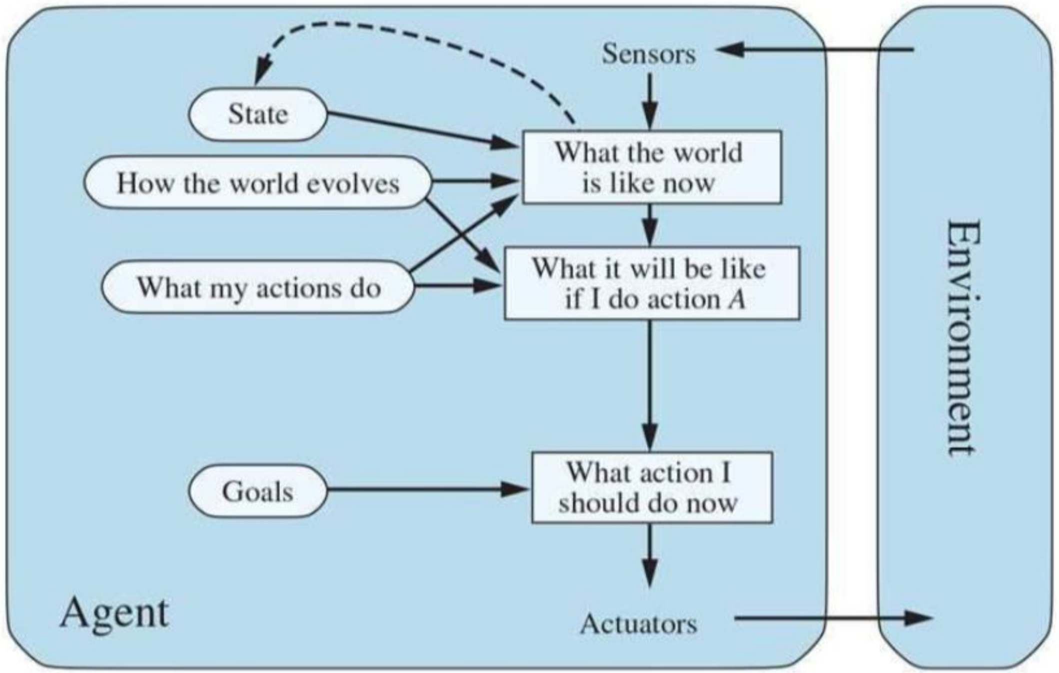
Knowing something about the current state of the environment is not always enough to decide what to do. A **goal-based agent** takes it a step further by using a goal in the future to help make decisions about It uses a specific method known as search and planning, meaning it targets the goal ahead and finds the right action in order to reach it.

For example, at a road junction, the taxi can turn left, turn right, or go straight on. The correct decision depends on where the taxi is trying to get to. In other words, as well as a current state description, the agent needs some sort of goal information that describes situations that are desirable—for example, being at a particular destination. The agent program can combine this with the model (the same information as was used in the model-based reflex agent) to choose actions that achieve the goal.

Example: A goal-based agent brakes when it sees brake lights because that’s the only action that it predicts will achieve its goal of not hitting other cars.

Although the goal-based agent appears less efficient, it is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified.

Goal-based agents Diagram


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

A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

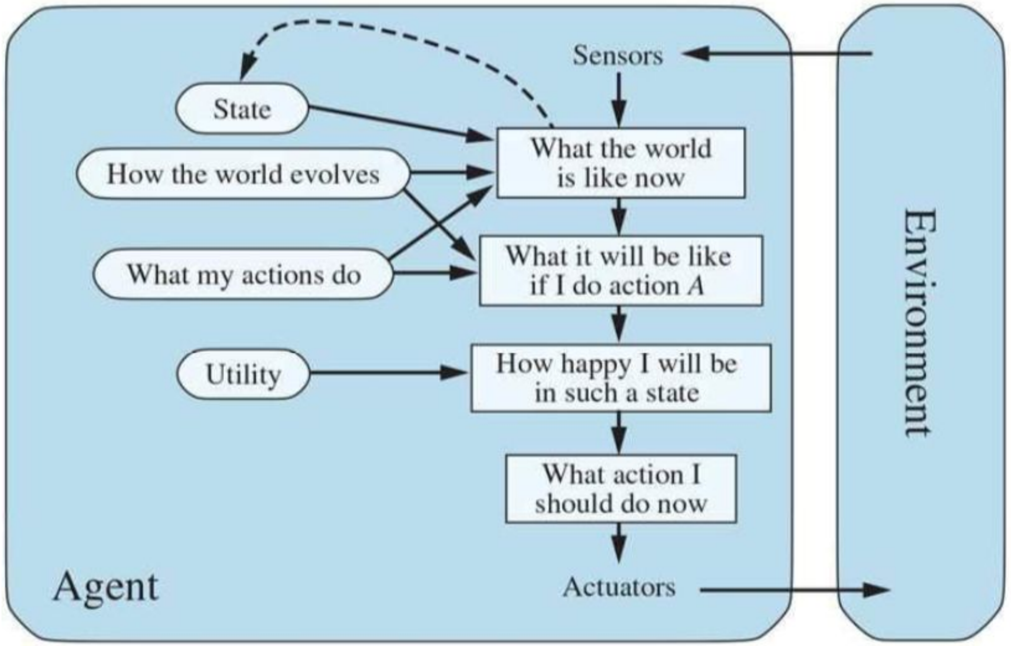
Utility-based agents

Goals alone are not enough to generate high-quality behavior in most environments. A **utility-based agent** is an agent that acts based not only on what the goal is, but the best way to reach that goal.

For example, many action sequences will get the taxi to its destination (thereby achieving the goal), but some are quicker, safer, more reliable, or cheaper than others. Goals just provide a crude binary distinction between “happy” and “unhappy” states. A more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent. Because “happy” does not sound very scientific, economists and computer scientists use the term utility instead.

An agent’s **utility function** is essentially an internalization of the performance measure. Provided that the internal utility function and the external performance measure are in agreement, an agent that chooses actions to maximize its utility will be rational according to the external performance measure.

Utility-based agents Diagram


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

A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

Learning Agents

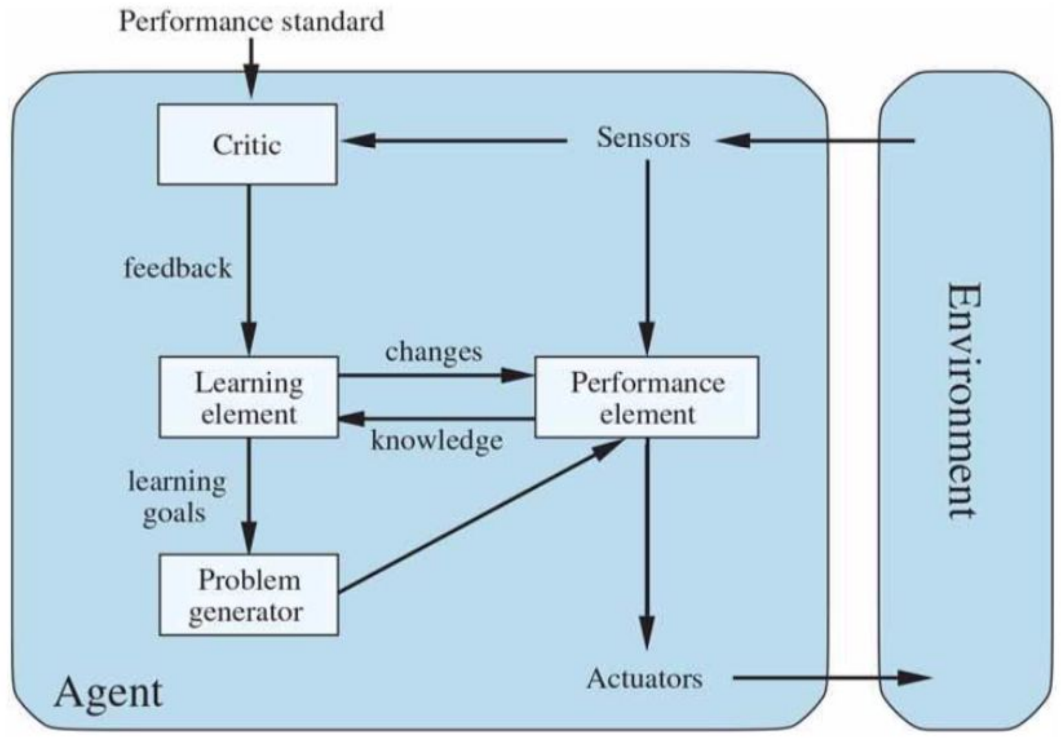
A **learning agent** in AI is the type of agent that can learn from its past experiences or it has learning capabilities. It starts to act with basic knowledge and then is able to act and adapt automatically through learning.

Turing (1950) considers the idea of actually programming his intelligent machines by hand. He estimates how much work this might take and concludes, “Some more expeditious method seems desirable.” The method he proposes is to build learning machines and then to teach them. In many areas of AI, this is now the preferred method for creating state-of-the-art systems. Any type of agent (model-based, goal-based, utility-based, etc.) can be built as a learning agent (or not).

A learning agent can be divided into four conceptual components:

1. The most important distinction is between the **learning element**, which is responsible for making improvements, and the performance element, which is responsible for selecting external actions.
2. The **performance element** is what we have previously considered to be the entire agent: it takes in percepts and decides on actions.
3. The learning element uses feedback from the **critic** on how the agent is doing and determines how the performance element should be modified to do better in the future.
4. **Problem Generator**: It is responsible for suggesting actions that will lead to new and informative experiences.

Learning Agents Diagram



A general learning agent. The "performance element" box represents what we have previously considered to be the whole agent program. Now, the "learning element" box gets to modify that program to improve its performance.