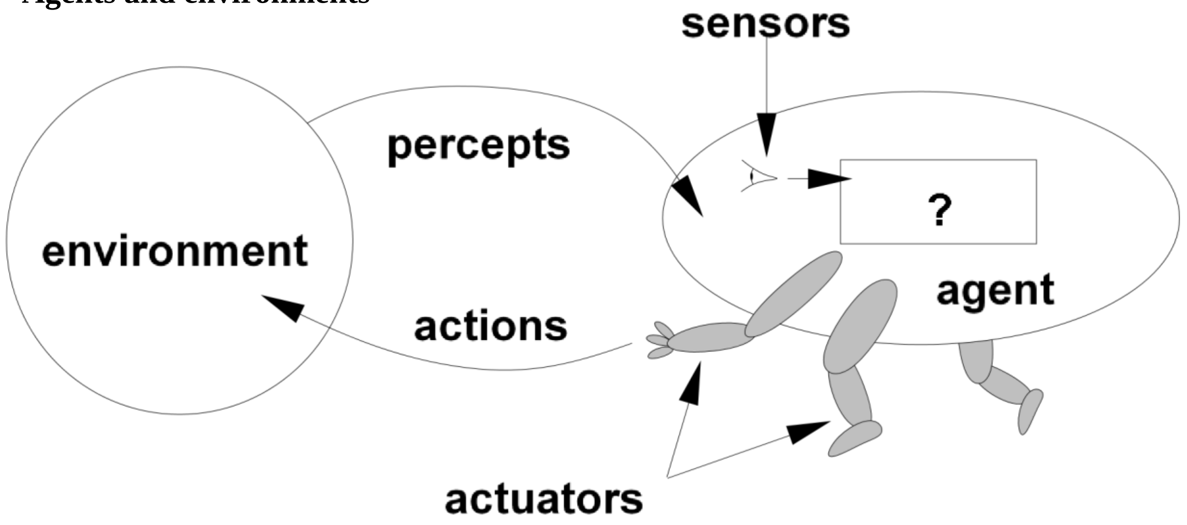
Unit 2 AI: Intelligent Agents

Agents

- An **"agent"** is anything that can be viewed as perceiving its environment through **sensors** and acting upon that environment through **actuators**.
- **Percept** refers to the agent's perceptual inputs at a given time instant; an agent's **perceptual sequence** is the <u>complete history</u> of everything the agent has ever perceived.
- In general, an agent's choice of action at any given instant can depend on the entire precept sequence observed to date, but not on anything it hasn't perceived.

Agents and environments



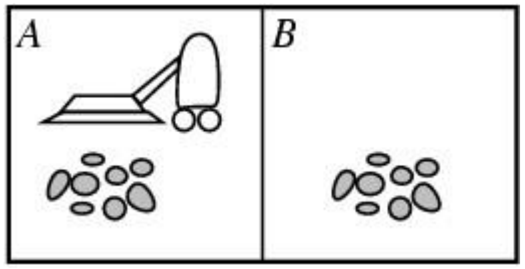
Agents include humans, robots, softbots(A computer program that acts on behalf of a user or another program), thermostats, etc. The agent function maps from percept histories to actions:

*

$$f : P \rightarrow A$$

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

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*
…	…

Rationality

- A rational agent is one that "does the right thing", i.e. the table for the agent function is filled out "correctly."
- But what does it mean to do the right thing? We use a **performance measure** to evaluate any given sequence of environment states.
- Importantly, we emphasize that the performance is assessed in terms of environment states <u>and not agent states</u>; self-assessment is often susceptible to self-delusion.
- Here is a relevant rule of thumb: *It is advisable to design performance measures according to what one actually wants in the environment, as opposed to how one believes that agent should behave*.
- What is **rational** at any given time depends on (at least) four things:
    - The performance measure
    - The agent's prior knowledge
    - The actions the agents can perform
    - The agent's percept sequence to date.

Definition of a **rational agent**: *For each possible precept 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 possesses.*

PEAS

To design a rational agent, we must specify the task environment

Consider, e.g., the task of designing an automated taxi:

- <u>Performance measure</u> safety, destination, profits, legality, comfort, ...
- <u>Environment</u> US streets/freeways, traffic, pedestrians, weather, ...
- <u>Actuators</u> steering, accelerator, brake, horn, speaker/display, ...
- <u>Sensors</u> video, accelerometers, gauges, engine sensors, keyboard, GPS, ...

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)

PEAS

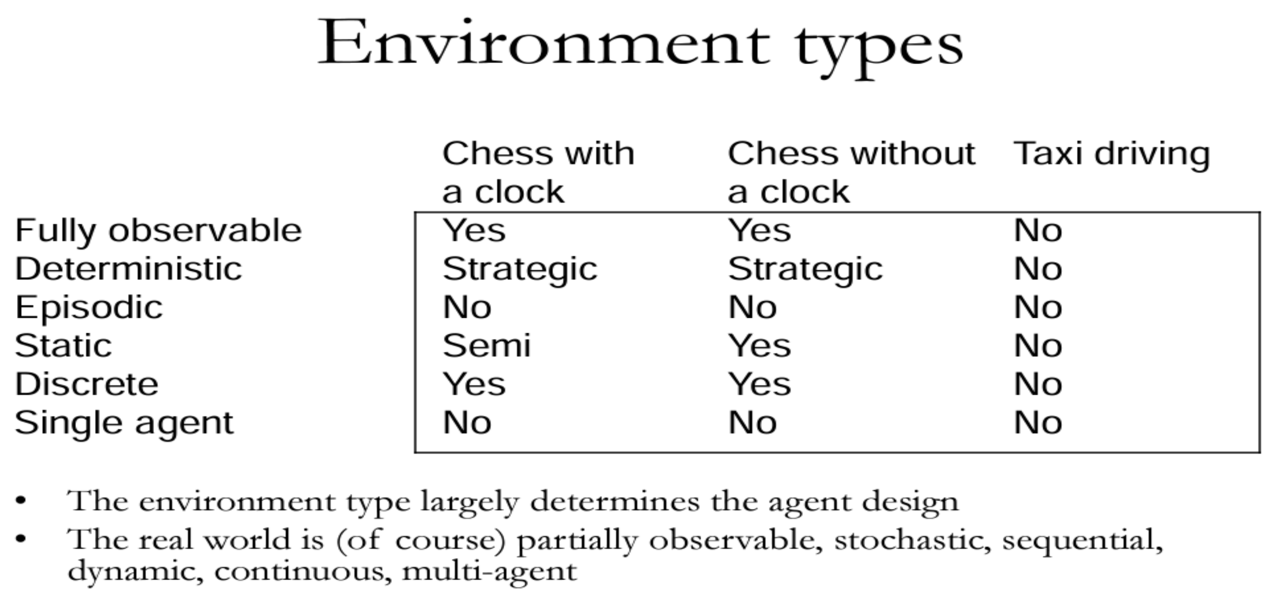
- **Agent**: Part-picking robot
- **Performance measure**: Percentage of parts in correct bins
- **Environment**: Conveyor belt with parts, bins
- **Actuators**: Jointed arm and hand
- **Sensors**: Camera, joint angle sensors

PEAS

- **Agent**: Interactive English tutor
- **Performance measure**: Maximize student's score on test
- **Environment**: Set of students
- **Actuators**: Screen display (exercises, suggestions, corrections)
- **Sensors**: Keyboard

Environment Types

- Fully observable (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.
- Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (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.
- Static (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is semidynamic if the environment itself does not change with the passage of time but the agent's performance score does)
- Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions.
- Single agent (vs. multiagent): An agent operating by itself in an environment.



Environment types

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

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

Structure of an AI Agent

The task of AI is to design an agent program which implements the agent function. The structure of an intelligent agent is a combination of architecture and agent program. It can be viewed as:

A**gent = Architecture + Agent program**

Following are the main three terms involved in the structure of an AI agent:

Architecture: Architecture is machinery that an AI agent executes on.

Agent Function: Agent function is used to map a percept to an action.

$$f : P^* \rightarrow A$$

Agent program: Agent program is an implementation of agent function. An agent program executes on the physical architecture to produce function f.

Agent types

Four basic types in order of increasing generality:

- simple reflex agents
- reflex agents with state
- goal-based agents
- utility-based agents

All these can be turned into learning agents

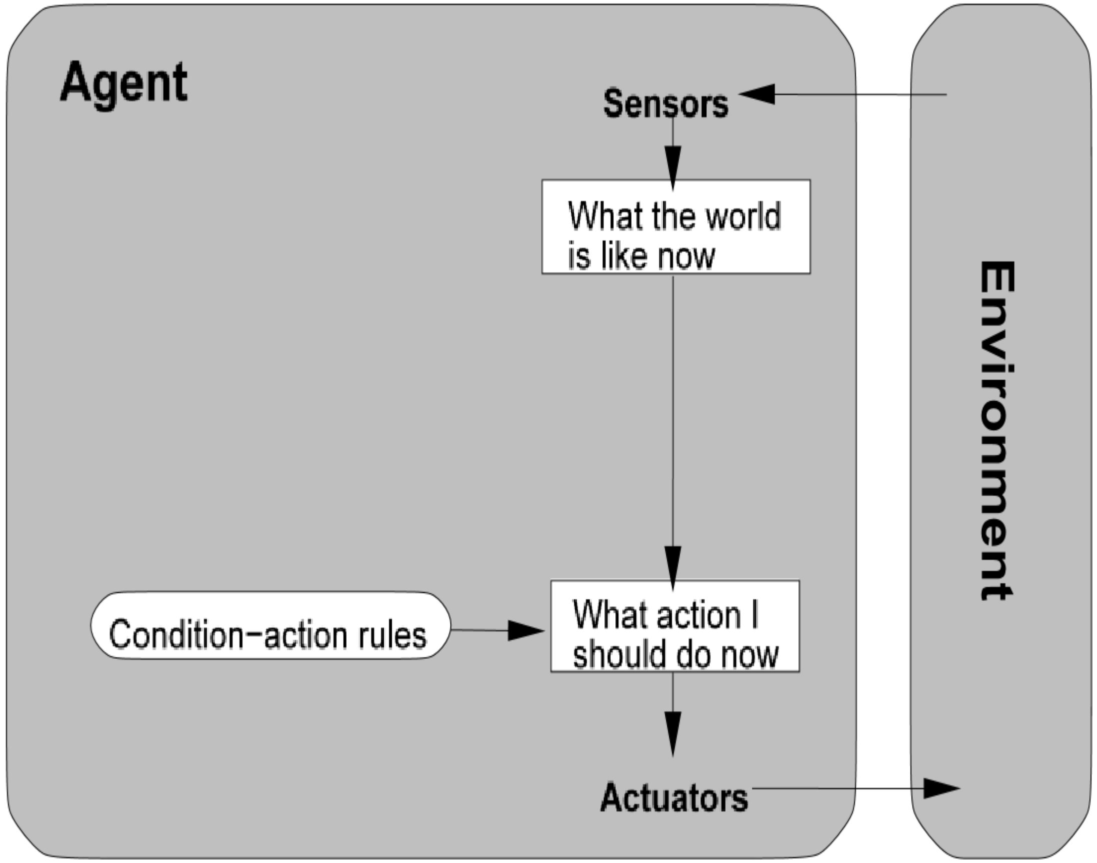
Simple Reflex Agents

The simplest kind of agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history. (An example for the vacuum world is below).

```
def perceive(percept):

 if percept == "dirty": return "clean"

elif percept == "clean": return "move"

else: return "do_nothing"

 # Example usage agent = SimpleReflexAgent() print(agent.perceive("dirty")) # Output: "clean"
```

```
print(agent.perceive("clean")) # Output: "move"
```

• Simple reflex agents are, naturally, simple, but they turn out to be of limited intelligence. The agent will only work if the correct decision can be made on the basis of only the current percept (so only if the environment is fully observable).



Model-based Reflex Agents

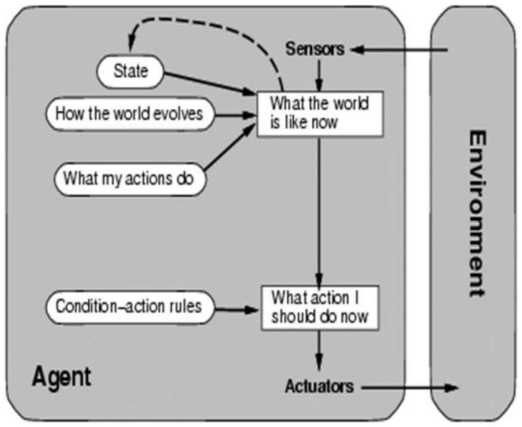
• The most effective way to handle partial observability is for the agent to keep track of the part of the world it can't see now; we say the agent maintains an internal state that depends on the percept history.

•Updating this internal state information requires two kinds of knowledge: (1) we need information about how the world evolves independent of the agent; (2) we need information about how the agent's own actions affect the world.

•The model of "how the world works" is called the model of the world.

•Note that we can't expect the representation to be perfect; instead the model of the world is an approximation – the agent's "best guess."

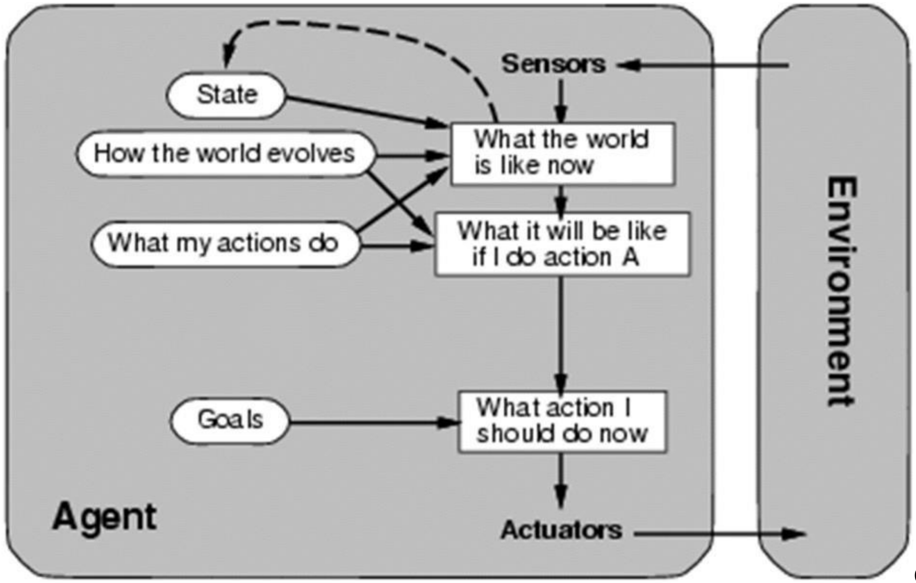
Model-based Reflex Agents



```
def perceive(self, percept):
        if percept == "dirty":
            self.state = "dirty"
            return "clean"
        elif percept == "clean" and self.state == "dirty":
            self.state = "clean"
            return "move"
        else: return "do_nothing
```

Goal-based Reflex Agents

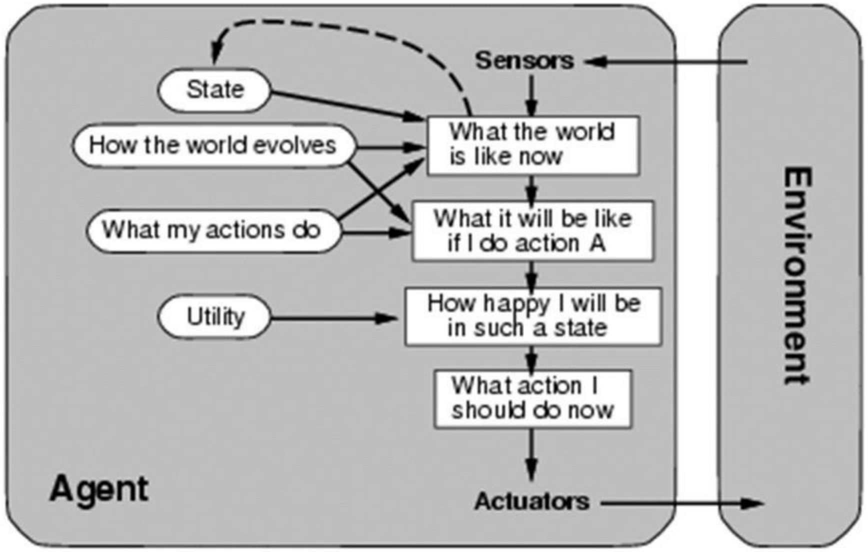
- Often, to make the correct decision, the agent needs some sort of goal information that describes situations that are desirable.
- Occasionally, goal-based action selection is straightforward (e.g. follow the action that leads directly to the goal); at other times, however, the agent must consider also search and planning. Decision making of this latter kind involves consideration of the future.
- Goal-based agents are commonly more flexible than reflex 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

```
def decide_action(self):
    if self.position < self.goal:
        return "move_forward"
    elif self.position > self.goal:
        return "move_backward"
    else:
        return "goal_reached"
```

Utility-based Reflex Agents

- Goals alone are not enough to generate high-quality behavior in most environments.
- An agent's **utility function** is essentially an international of the performance measure. If the internal utility function and the external performance measure are in agreement, then an agent that <u>chooses actions to maximize its utility</u> will be rational according to the external performance measure.
- A utility-based agent has many advantages in terms of flexibility and learning (e.g. in the case of <u>conflicting goals</u> and cases when <u>there exist several goals</u>).


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

```
def decide_action(self):
    if self.energy > 20:
        return "explore"
    elif self.satisfaction < 50:
        return "rest"
    else:
        return "do_nothing"
```

Learning Agents

• A learning agent is comprised of (4) components:

1. the **learning element**, which is responsible for making improvements;
2. the **performance element**, which is responsible for selecting external actions;
3. the **critic**, which gives feedback to the agent, and determines how the performance should be modified;
4. the **problem generator** is responsible for suggesting actions that will lead to new and informative experiences.

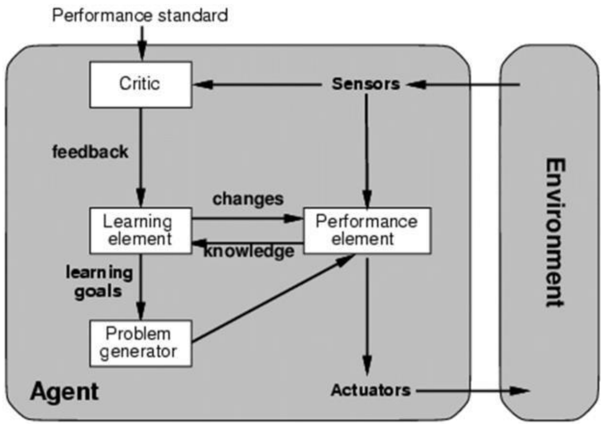
<u>Consider the taxi example</u>:

Performance element: whatever collection of knowledge and procedures the taxi has for selecting its driving actions.

Learning element: Formulates a rule based on experience

Critic: Adds new rules, based on feedback.

Problem Generator: Identify certain areas of behavior in need of improvement and suggest experiments.



Agent Type	Decision Basis	Complexity	Example Application
Simple Reflex	Current percept only	Low	Thermostat
Model-Based Reflex	Current percept + state	Medium	Robot vacuum cleaner
Goal-Based	Goal-driven	High	GPS navigation
Utility-Based	Utility maximization	High	Autonomous vehicle
Learning	Experience-based	Variable	Recommendation systems, self-improving bots