Unit -2: Intelligent Agents

Introduction of Agents

- Agent: somethings that acts.
- An **agent** is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators. Hence, an agent gets percepts one at a time, and maps this percept sequence to actions.
- **A rational agent** is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome
- Example of agents :

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

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

  A software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets.
- **Intelligent agent** is one that is capable of doing following things autonomously
  - Perceive environment respond to change that occur in order to satisfy design objective
  - Interact with other agent and to satisfy design objective.
  - Needs to be goal directed as well as reactive
- Intelligent agent means it does things based on reasoning, while rational agent means it does the best action (or reaction) for a given situation.

Properties of an agent

- Autonomous
- Interacts with other agents plus the environment
- Reactive to the environment
- Pro-active (goal- directed)

Configuration of Agent

An agent consists of **sensors** to perceive environment and **actuators** to act upon the environment.

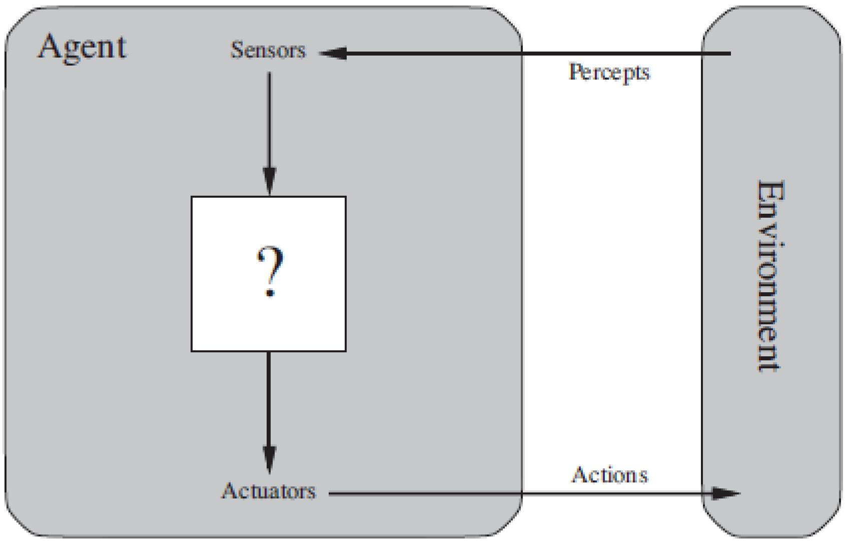

Figure: Agents interact with environments through sensors and actuators

Agent perceives its environment through sensors and acts upon the environment through actuators.

For Human agent

Sensors: Eyes (vision), ears (hearing), skin (touch), tongue (gestation), nose (olfaction), neuromuscular system (proprioception)

Percepts:

• At the lowest level – electrical signals from these sensors

• After preprocessing – objects in the visual field (location, textures, colors, …), auditory streams (pitch, loudness, direction), …

Actuators / Effectors: limbs, digits, eyes, tongue, …..

Actions: lift a finger, turn left, walk, run, carry an object, …

For a Robotic agent

Sensors : Camaera, microphone, IR range finder,..

Actuators : Motor,…

Action: Steering, displaying information, …

PEAS Description

- In designing an agent, the first step must always be to specify the task environment as fully as possible
- Task environment is specified by PEAS(**P**erformance, **E**nvironment, **A**ctuators, **S**ensors) description

PEAS description for Automated Taxi's task environment:

Performance: Safe, fast, legal, comfortable trip, maximize profits

Environment: Roads, other traffic, pedestrians, customers

Actuators: Steering, accelerator, brake, signal, horn, display

Sensors: Cameras, sonar, speedometer, GPS, odometer, accelerometer,engine sensors, keyboard

PAGE Description

- An agent and its task environment can be described using **PAGE** description where **P** stands for Percepts, **A** for Actions, **G** stands for Goals and **E** means Environment.

PAGE description for Automated Taxi:

Percepts: Video, sonar, speedometer, odometer, engine sensors, keyboard input, microphone, GPS, …

Actions: Steer, accelerate, brake, horn, speak/display, …

Goals: Maintain safety, reach destination, maximize profits (fuel, tire wear), obey laws, provide passenger comfort, …

Environment: Urban streets, freeways, traffic, pedestrians, weather, customers, …

Percept and Percept Sequence

Percept: The Agent's perceptual inputs at any given instant.

Percept Sequence: The complete history of everything the agent has ever perceived.

In general, *an agent's choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn't perceived.*

Agent Function and Agent Program

- Mathematically speaking, we say that an agent's behavior is described by the agent function that maps any given percept sequence to an action.
- The **agent function** is mathematical concept that maps percept sequence to actions

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

- The agent function will internally be represented by the **agent program**.
- The **agent program** is concrete implementation of agent function it runs on the physical architecture to produce **f.**
- The agent function is an abstract mathematical description; the agent program is a concrete implementation, running within some physical system.

The vacuum-cleaner world: Example of an Agent

Environment: square A and B

Percepts: [location and content] *E.g. [A, Dirty]*

Actions: left, right, suck, and no-op

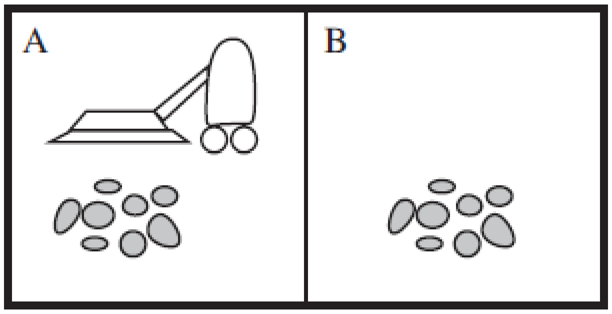

Figure: A vacuum-cleaner world with just two locations.

Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure

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*
⋮	⋮

The Concept of Rationality

- A rational agent is one that does the right thing

  —conceptually speaking, every entry in the table for the agent function is filled out correctly.

- Obviously, doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing?

  - Right action is the one that will cause the agent to be most successful.

- Therefore we need some way to measure success of an agent. Performance measures are the criterion for success of an agent behavior.
- There is not one fixed performance measure for all tasks and agents; typically, a designer will devise one appropriate to the circumstances

  E.g., performance measure of a vacuum-cleaner agent could be

  - amount of dirt cleaned up,
  - amount of time taken,
  - amount of electricity consumed,
  - amount of noise generated, etc.
- It is not easy task to choose the performance measure of an agent.

  For example if the performance measure for automated vacuum cleaner is "The amount of dirt cleaned within a certain time" Then a rational agent can maximize this performance by cleaning up the dirt , then dumping it all on the floor, then cleaning it up again , and so on. Therefore "How clean the floor is" is better choice for performance measure of vacuum cleaner.

  As a general rule, it is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave.

- What is rational at a given time depends on four things:
    1. The performance measure that defines 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 to date.

  This leads to a **definition of a rational agent**:

  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.

- A rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date and prior environment knowledge.

Environments and its properties

- To design a rational agent we must specify its task environment.
- Task environments are essentially the "problems" to which rational agents are the "solutions."
- The flavor of the task environment directly affects the appropriate design for the agent program.
- In designing an agent, the first step must always be to specify the task environment as fully as possible
- Task environment is specified by PEAS(**P**erformance, **E**nvironment, **A**ctuators, **S**ensors) description

  PEAS description for Automated Taxi's task environment:

    Performance: Safe, fast, legal, comfortable trip, maximize profits

    Environment: Roads, other traffic, pedestrians, customers

    Actuators: Steering, accelerator, brake, signal, horn, display

    Sensors: Cameras, sonar, speedometer, GPS, odometer, accelerometer,engine sensors, keyboard

Properties of Environment (types/classes of Environment)

Following are the dimensions based on which the environments can be categorized.

- **Fully observable** vs. **partially observable**
- **Single agent** vs. **multiagent**
- **Deterministic** vs. **stochastic**
- **Episodic** vs. **sequential**
- **Static** vs. **dynamic**
- **Discrete** vs. **continuous**
- **Known** vs. **unknown**

Fully observable vs. partially observable

- If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is **fully observable.**
- Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world.
- An environment might be **partially observable** because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data—for example, a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares, and an automated taxi cannot see what other drivers are thinking.
- If the agent has no sensors at all then the environment is unobservable.

Single agent vs. multiagent

- Environment in which only one agent is acting is called **single agent** environment whereas the environment in which more than one agents are acting is called multiagent environment.
- For example, an agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two-agent environment.
- Multi-agent environment can be:

- **Competitive:** For example, in chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's performance measure. Thus, chess is a **competitive multiagent** environment
- **Cooperative**: In the taxi-driving environment, on the other hand, avoiding collisions maximizes the performance measure of all agents, so it is a partially **cooperative multiagent** environment.

<u>**Deterministic** vs. **stochastic**</u>

- If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is **deterministic**; otherwise, it is **stochastic**
- In principle, an agent need not worry about uncertainty in a fully observable, deterministic environment
- If the environment is partially observable, however, then it could appear to be stochastic
- Most real situations are so complex that it is impossible to keep track of all the unobserved aspects; for practical purposes, they must be treated as stochastic
- *Taxi driving* is clearly stochastic in this sense, because one can never predict the behavior of traffic exactly; moreover, one's tires blow out and one's engine seizes up without warning.
- *The vacuum world* as we described it is deterministic, but variations can include stochastic elements such as randomly appearing dirt and an unreliable suction mechanism
- The word "stochastic" generally implies that uncertainty about outcomes is quantified in terms of probabilities; a **nondeterministic** environment is one in which actions are characterized by their possible outcomes, but no probabilities are attached to them.
- **Nondeterministic** environment descriptions are usually associated with performance measures that require the agent to succeed for all possible outcomes of its actions.

Episodic vs. **sequential**

- In an **episodic** task environment, the agent's experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action.
- In episodic environments, the choices of action in each episode depends only on episode itself i.e., the next episode does not depend on the actions taken in previous episodes.
- Many classification tasks are episodic.
- For example, an agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions; moreover, the current decision doesn't affect whether the next part is defective.
- In **sequential** environments, the current decision could affect all future decisions
- *Chess* and *taxi driving* are sequential: in both cases, short-term actions can have long-term consequences.
- Episodic environments are much simpler than sequential environments because the agent does not need to think ahead.

Static vs. **dynamic**

- If the environment can change while an agent is deliberating, then we say the environment is **dynamic** for that agent; otherwise, it is **static**
- Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time.
- Dynamic environments, on the other hand, are continuously asking the agent what it wants to do; if it hasn't decided yet, that counts as deciding to do nothing
- If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is semidynamic.
- *Taxi driving* is clearly dynamic: the other cars and the taxi itself keep moving while the driving algorithm dithers about what to do next.
- *Chess*, when played with a clock, is semidynamic.
- *Crossword puzzles* are static.

<u>Discrete</u> <u>vs.</u> **<u>continuous</u>**

- The discrete/continuous distinction applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent
- For example, the chess environment has a finite number of distinct states (excluding the clock). Chess also has a discrete set of percepts and actions.
- Taxi driving is a continuous-state and continuous-time problem: the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and do so smoothly over time. Taxi-driving actions are also continuous (steering angles, etc.). Input from digital cameras is discrete, strictly speaking, but is typically treated as representing continuously varying intensities and locations.

<u>Known</u> <u>vs.</u> **<u>unknown</u>**

- This distinction refers not to the environment itself but to the agent's (or designer's) state of knowledge about the "laws of physics" of the environment.
- In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given
- If the environment is unknown, the agent will have to learn how it works in order to make good decisions
- Note that the distinction between known and unknown environments is not the same as the one between fully and partially observable environments. It is quite possible for a known environment to be partially observable—for example, in solitaire card games, I know the rules but am still unable to see the cards that have not yet been turned over. Conversely, an unknown environment can be fully observable—in a new video game, the screen may show the entire game state but I still don't know what the buttons do until I try them.

Note: As one might expect, the hardest case is *partially observable*, *multiagent*, *stochastic*, *sequential*, *dynamic*, *continuous*, and *unknown*.

Agent structures

- 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 **architecture**

agent* = *architecture* + *program

- The program we choose has to be one that is appropriate for the architecture.
  If the program is going to recommend actions like Walk, the architecture had better have legs.
- The architecture might be just an ordinary PC, or it might be a robotic car with several onboard computers, cameras, and other sensors.
- In general, the architecture makes the percepts from the sensors available to the program, runs the program, and feeds the program's action choices to the actuators as they are generated

Types of Agent (Agent Programs) :

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

Simple reflex agents

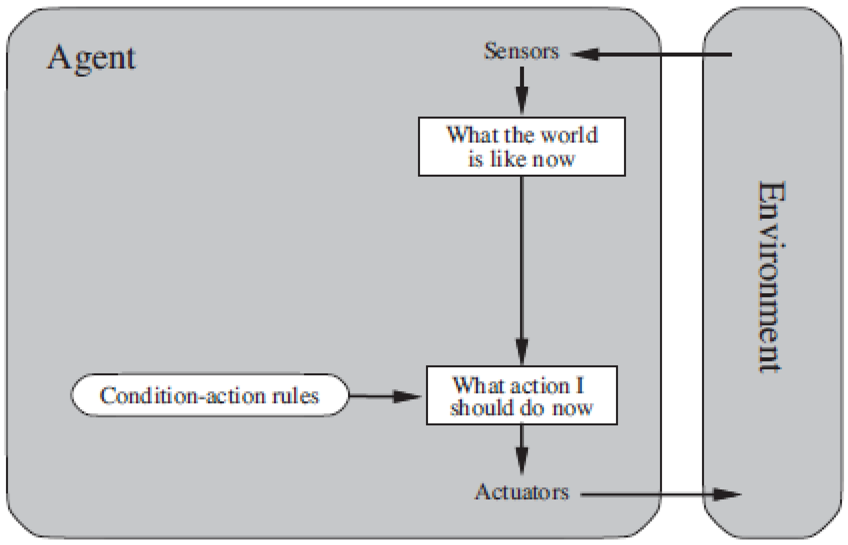

Figure: Simple reflex agent.

- 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 : its decision is based only on the current location and on whether that location contains dirt.
- Based on condition–action rule (if-then rule / situation–action rule)
    - **if** *car-in-front-is-braking* **then** *initiate-braking*
- Simple reflex agents have the admirable property of being simple, but they turn out to be of limited intelligence
- They will work only if the correct decision can be made on the basis of only the current percept—that is, 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.
- That is, the agent should maintain some sort of internal state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state
- Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program.
    - First, we need some information about how the world evolves independently of the agent—for example, that an overtaking car generally will be closer behind than it was a moment ago.
    - Second, we need some information about how the agent's own actions affect the world—for example, that when the agent turns the steering wheel clockwise, the car turns to the right, or that after driving for five minutes northbound on the freeway, one is usually about five miles north of where one was five minutes ago
- This knowledge about "how the world works"—whether implemented in simple Boolean circuits or in complete scientific theories—is called a **model** of the world.
  An agent that uses such a model is called a model-based agent.

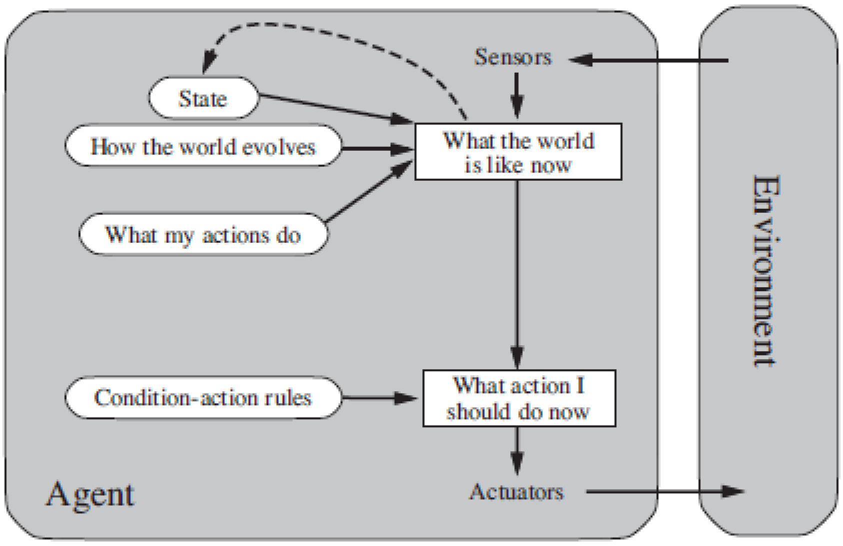

Figure: A model-based reflex agent.

Goal-based agents

- ❖ Knowing something about the current state of the environment is not always enough to decide what to do. 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 the passenger's 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.
- ❖ Goal- based agent 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.
- ❖ Sometimes goal-based action selection is straightforward—for example, when goal satisfaction results immediately from a single action. Sometimes it will be more tricky—for example, when the agent has to consider long sequences of twists and turns in order to find a way to achieve the goal.
- ❖ Search and planning are used to find action sequences that achieve the agent's goals.

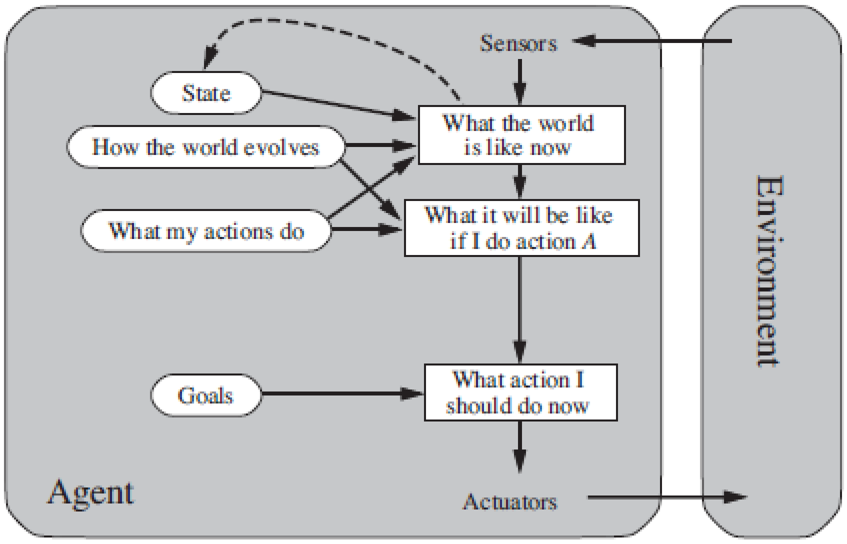
Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

Figure: Goal-based agent

Utility-based agents

→ Goals alone are not enough to generate high-quality behavior in most environments. 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

→ The word “utility” here refers to “the quality of being useful”.

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

→ A rational utility-based agent chooses the action that maximizes the expected utility of the action outcome.

→ 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

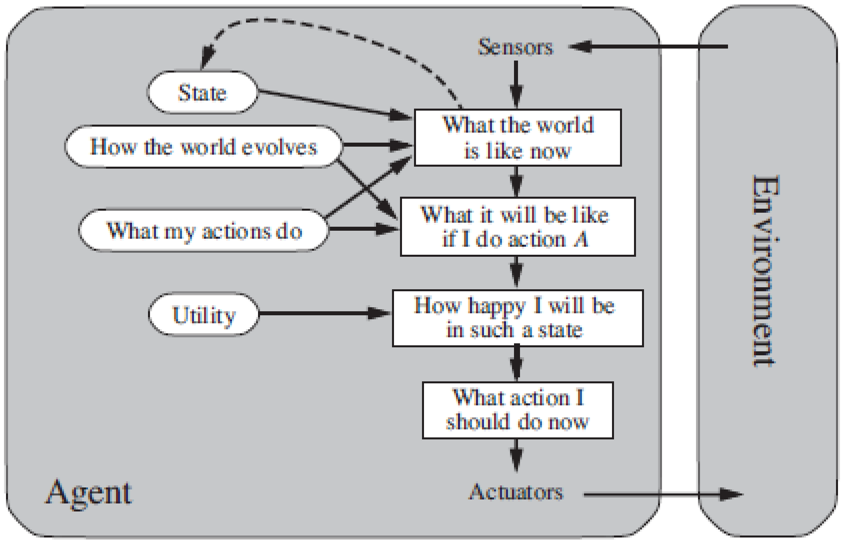
Sensors
State
How the world evolves
What my actions do
Utility
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Actuators
Agent
Environment

Figure: Utility-based agent

Learning agents

⇒ Agents that can learn

⇒ In many areas of AI, this is now the preferred method for creating state-of-the-art systems.

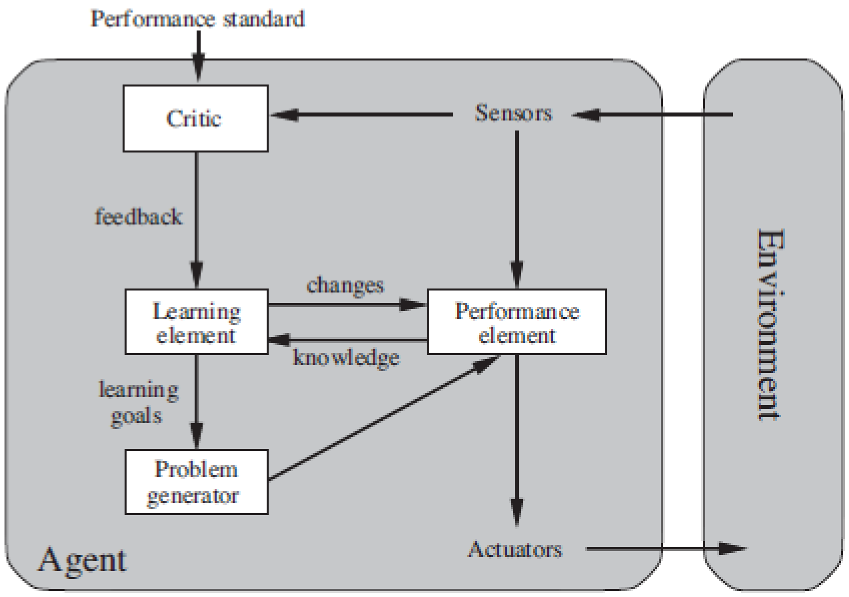

Figure: A general learning agent

⇒ Such agent operates in initially unknown environments and becomes more competent than its initial knowledge alone might allow.

⇒ Learning element is one of the elements of learning agent which is responsible for making improvements, and the other element, performance element, is responsible for selecting external actions. Performance element takes in percepts and decides on actions.

⇒ 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.

⇒ Problem generator is responsible for suggesting actions that will lead to new and informative experiences.

⇒ Learning in intelligent agents can be summarized as a process of modification of each component of the agent to bring the components into closer agreement with the available feedback information, thereby improving the overall performance of the agent.