Artificial Intelligence

CHAPTER 2: **Intelligent Agent**

Sourav Kumar Giri
Asst. Professor

School of Computer Engineering
KIIT Deemed to be University

Chapter Outline

- Agent & Environment
- The good behavior: The concept of rationality
- The nature of Environments
- The Structure of Agents
- The Learning Agent

Agent

- An agent is anything that can be viewed as perceiving its environment through **sensors** and acting upon that environment through **actuators**.
- Examples of Agent-
  - A **human agent** has eyes, ears, and other organs for sensors and hands, legs, mouth, and other body parts 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.

Agent's Perception & Action

- **Percept-**refers to an agents perceptual input at any given instant.
- **Percept sequence-** refers to the complete history of what an agent has perceived so far.
- **Action-** an agent's choice of action at any given instant can depend on the entire percept sequence observed till that time.
- **Agent function-** an agent's behavior is described by the agent function that maps any percept sequence to an action.

Agent & Environment

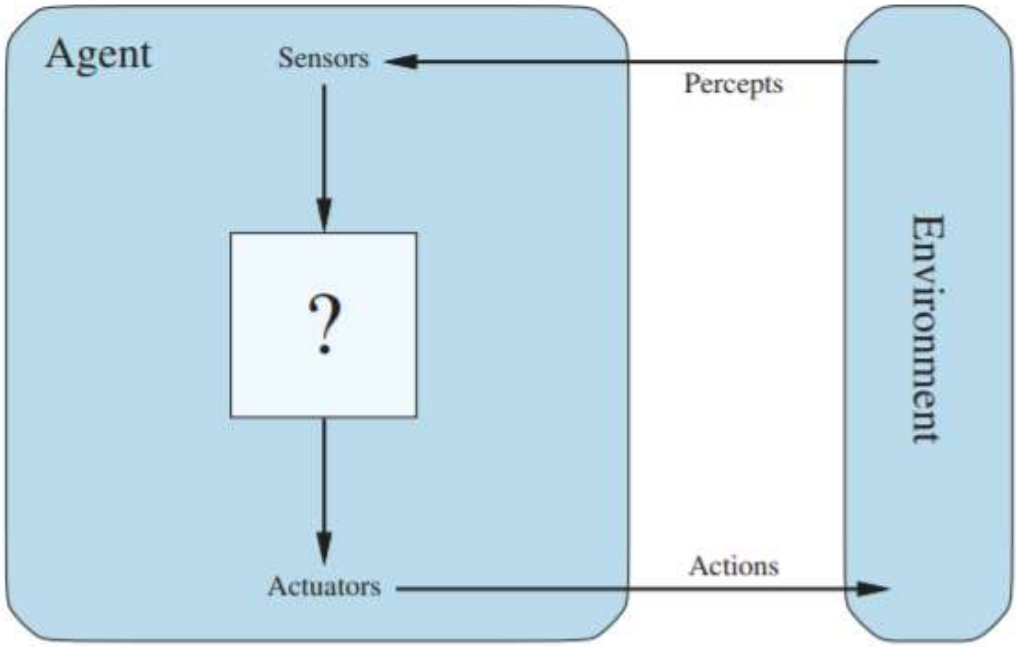
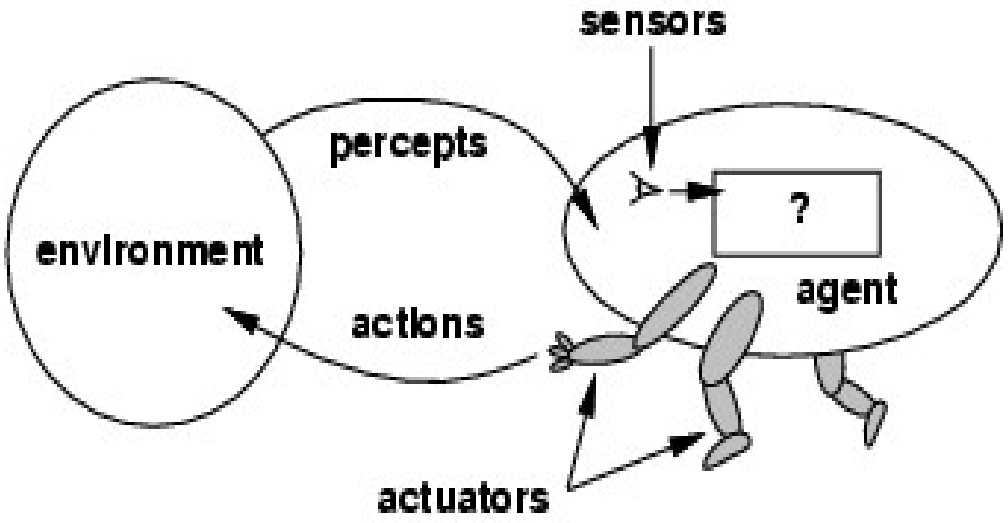


Fig. Agents interact with environments through sensors and actuators

Agent & Environment

- The agent function maps from percept histories to actions
- The agent function is implemented by an agent program.
- The agent program runs on the physical architecture to produce *f*
- **agent = architecture + program**, *where architecture is composed of senor, actuator and the physical/logical environment*



Vacuum-cleaner world

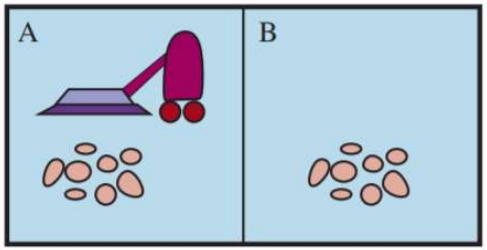


Fig. A vacuum-cleaner world with just two locations

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*
⋮	⋮

Table. Partial tabulation of a simple agent function for the vacuum-cleaner world

The concept of Rationality

- A ***rational agent*** is one that does the ***right thing*** based on what it can perceive and the actions it can perform.
- **Definition-** 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.
- Degree of rationality at any given time depends on four things:
  - *The performance measure that defines the criterion of success.*
  - *The agent's prior knowledge of the environment.*
  - *The actions that the agent can perform.*
  - *The agent's percept sequence to date.*

The concept of Rationality

- **Performance measure-** An objective criterion for success of an agent's behavior.

  Example- performance measure of a vacuum-cleaner agent could be

  - the amount of dirt cleaned up
  - the amount of time taken
  - the amount of electricity consumed
  - the amount of noise generated, etc .

- **Requirements for rationality-** information gathering capability, learning ability, ability to adapt, autonomy etc.

Omniscience, learning, and autonomy

- An ***omniscient agent*** knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality.
- Rationality maximizes ***expected performance***, while perfection maximizes ***actual performance***.
- A rational agent not only to gather information but also to ***learn*** as much as possible fro m what it perceives.
- The agent's initial configuration could reflect some prior knowledge of the environment, but as the agent gains experience this may be modified and augmented.
- A rational agent should be autonomous—it should learn what it can to compensate for partial or incorrect prior knowledge.

Building intelligent agent

- An agent is completely specified by the agent function which maps percept sequences to action.
- An agent has some internal data structures that is updated as and when new percepts arrive.
- The data structures are operated on by the agent's decision making procedures to generate an action choice, which is then passed to the architecture to get executed.
- An agent consists of an architecture plus a program that runs on that architecture.
- In designing intelligent systems , following factors are considered-
  - **Percepts-** the inputs to our system
  - **Actions-** the outputs of our system
  - **Goals-** what the agent is expected to achieve
  - **Environment-** what the agent is interacting with

Specifying the task environment (PEAS)

- Task environments are problems to which rational agents are the solutions. The task environment is specified by PEAS.
- **PEAS- P**erformance measure, **E**nvironment, **A**ctuators, **S**ensors.
- PEAS specifies the settings for an intelligent agent design.

Agent Type	Performance measure	Environment	Actuators	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, engine sensors, key board

PEAS elements for different types of agent

Agent Type	Performance measure	Environment	Actuators	Sensors
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
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display categorization of scene	Color pixel Arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Maximize purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Maximize student's score on test	Set of students, testing agency	Display exercises, suggestions, corrections	Keyboard entry

Self driving car

Part picking robot

Types of task environment

❑ Fully observable vs Partially observable environment-

- ❑ **Fully observable environment-** agent's sensors give it access to the complete state of the environment at each point in time.

  Example: Cross word puzzle, Chess with a clock, Image analysis

- ❑ **Partially observable environment-** due to noisy and inaccurate sensors because parts of the state are simply missing from the sensor data.

  Example: 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.

❑ Single vs Multi-agent environment

- ❑ **Single agent environment**- Environment involving one agent.

  Example- cross-word puzzle.

- ❑ **Multi-agent environment**- Environment involving more than one agents.
  Example- Chess game.

Types of task environment

- **Deterministic vs Stochastic environment**
  - **Deterministic environment-** the next state of the environment is completely determined by the current state and the action executed by the agent.

    Example: Cross word puzzle
  - **Stochastic environment-** the next state of the environment is not completely determined by the current state and the action executed by the agent.

    Example: Automated taxi driver
- **Episodic vs Sequential environment**
  - **Episodic environment**- the agent's experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action. The next episode does not depend on the actions taken in previous episodes.

    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.
  - **Sequential environment**- the current decision could affect all future decisions

    Example- Chess and Taxi driving are sequential: in both cases, short-term actions can have long-term consequences.

Types of task environment

- **Static vs Dynamic environment**
  - **Static environment-** the environment can not change while an agent is deliberating.

    Example- Cross word puzzle
  - **Dynamic environment-** the environment can change while an agent is deliberating.

    Example- Automated taxi driving
  - **Semi dynamic environment-** If the environment itself does not change with the passage of time but the agent's performance score does

    Example- Chess game, when played with a clock
- **Discrete vs Continuous environment**
  - **Discrete environment**- state of the environment is handled in discrete manner.

    Example- Chess game
  - **Continuous environment**- state of the environment is handled in continuous manner.

    Example- taxi driving

Examples of task environment

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

Agent programs

- Agent programs take the current percept as input from the sensors and return an action to the actuators.
  - ***agent program***, which takes the current percept as input, and the ***agent function***, which takes the entire percept history as input.
- **Table driven agent program**

function TABLE-DRIVEN-AGENT(percept) returns an action
static: percepts, a sequence, initially empty
table, a table of actions, indexed by percept sequences, initially fully specified
append percept to the end of percepts
action→LOOKUP(percepts, table)
return action

Table driven agent program

- Designers must construct a table that contains the appropriate action for every possible percept sequence.
- The lookup table will contain $\sum_{t=1}^{T} |P|^t$ no of entries.
  - Where **P** is the set of possible percepts and **T** is the life time of the agent (the total no of percepts it will receive)
- The visual input from a single camera comes in at the rate of roughly 27 megabytes per second (30 frames per second, 640×480 pixels with 24 bits of color information). This gives a lookup table with over $10^{250000000000}$ entries for an hour's driving.
- the lookup table for chess have at least 10^{150} entries.

Table driven agent program: A Failure

- The daunting size of these tables means that
  - no physical agent in this universe will have the space to store the table
  - the designer would not have time to create the table
  - no agent could ever learn all the right table entries from its experience
  - even if the environment is simple enough to yield a feasible table size, the designer still has no guidance about how to fill in the table entries.

Vacuum Cleaner Example

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*

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 Program

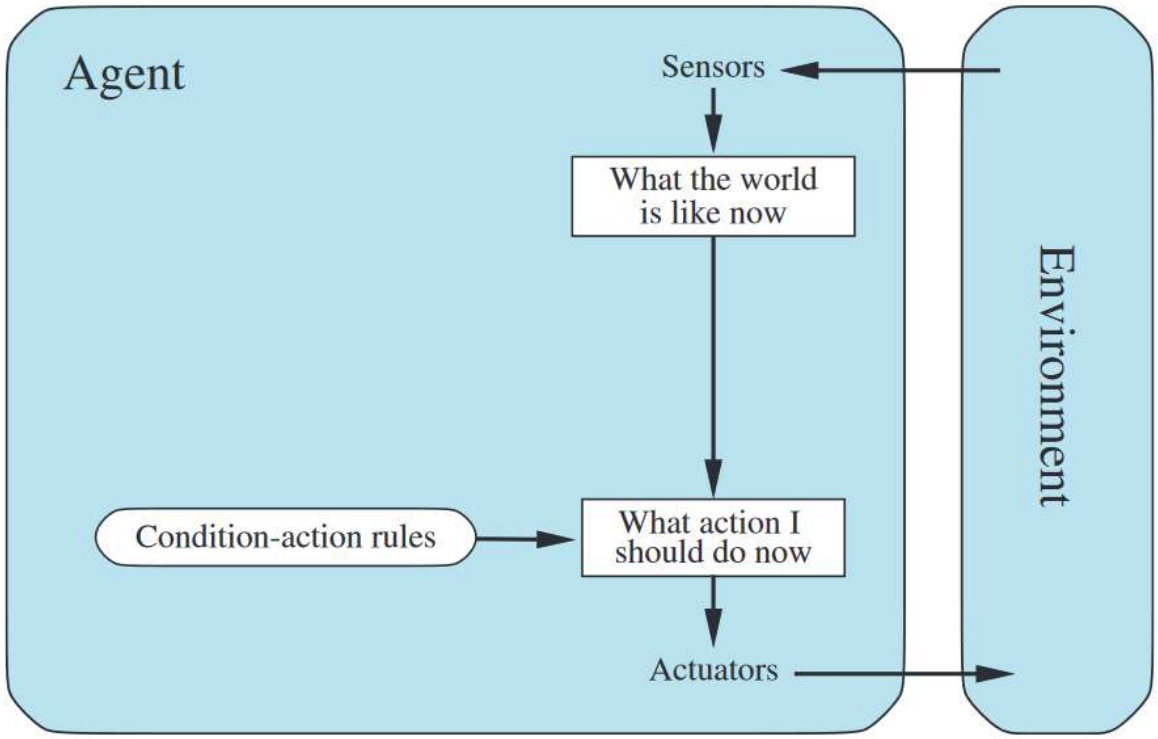
- Four basic types of agents in order of increasing sophistication:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

Simple reflex agents

- This is a simple type of agent which works on the basis of current percept and not based on the rest of the percepts history.
- The agent function, in this case, is based on condition-action rule where the condition or the state is mapped to the action such that action is taken only when condition is true or else it is not e.g.

  if car-in-front-is-braking then initiate-braking.
- If the environment associated with this agent is fully observable, only then is the agent function successful, if it is partially observable, in that case the agent function enters into infinite loops that can be escaped only on randomization of its actions.
- The problems associated with this type of agent include very limited intelligence, No knowledge of non-perceptual parts of the state, huge size for generation and storage and inability to adapt to changes in the environment.

Simple reflex agents



Schematic diagram of a simple reflex agent

Simple reflex agents

- Simple reflex agent program-

function SIMPLE-REFLEX-AGENT (percept) **returns** an action

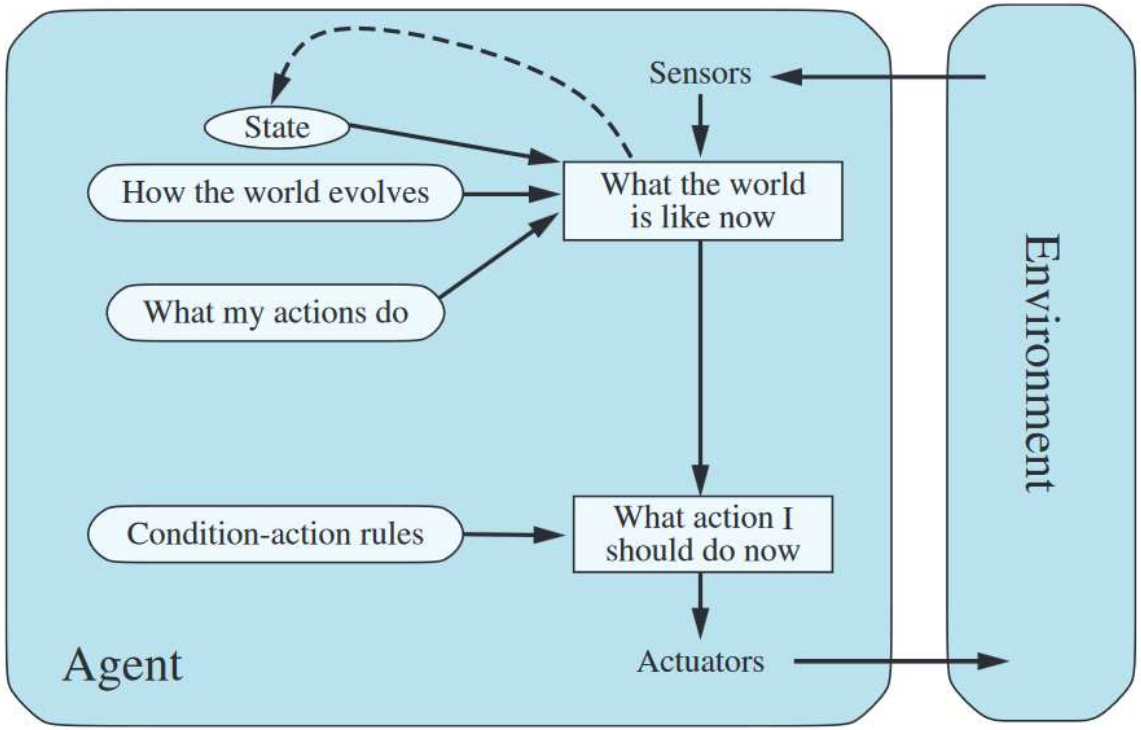
persistent: rules, a set of condition–action rules

state ← **INTERPRET-INPUT** (percept)
rule ← **RULE-MATCH** (state, rules)
action ← rule. **ACTION**
return action

Model-based reflex agents

- Model-based reflex agent can handle partially observable environments by tracking the situation and using a particular model related to the world.
- It consists of two important factors, which are Model and Internal State.
- Model provides knowledge and understanding of the process of occurrence of different things in the surroundings such that the current situation can be studied and a condition can be created. Actions are performed by the agent based on this model.
- Internal State uses the perceptual history to represent a current percept. The agent keeps a track of this internal state and is adjusted by each of the percepts. The current internal state is stored by the agent inside it to maintain a kind of structure that can describe the unseen world.
- The state of the agent can be updated by gaining information about how the world evolves and how the agent's action affects the world.

Model-based reflex agents



Schematic diagram of a model-based reflex agent

Model-based reflex agent program

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)

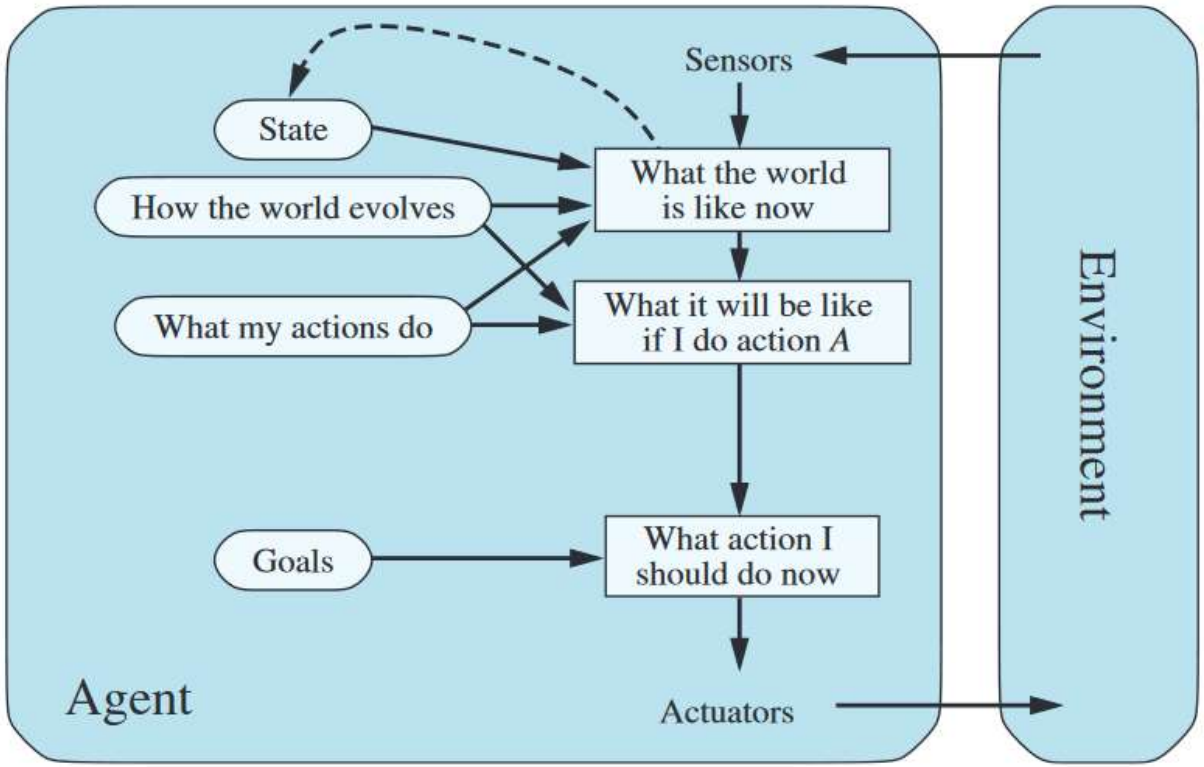
action ← *rule*.ACTION

return *action*

Goal-based agent

- Knowing about the current state of the environment is not always enough to decide what to do; additionally sort of goal information which describes situations that are desirable is also required. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state.
- Current state of the environment is always not enough
- The goal is another issue to achieve
  - Judgment of rationality / correctness
- Actions chosen→ goals, based on
  - the current state
  - the current percept

Goal-based agent



Schematic diagram of a goal-based reflex agent

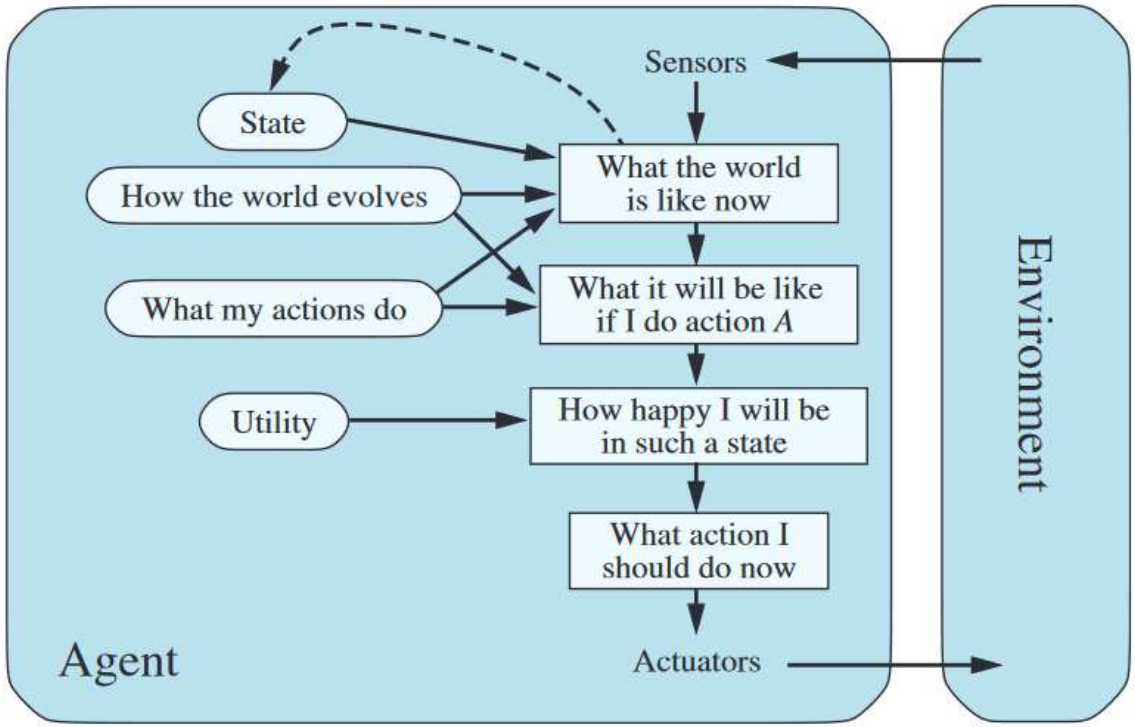
Utility-based agent

- Goal based agents only distinguish between goal states and non-goal states.
- 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.

Utility-based agent

- It is possible to define a measure of how desirable a particular state is. This measure can be obtained through the use of a utility function which maps a state to a measure of the utility of the state.
- Utility function maps a state onto a real number, which describes the associated degree of happiness.

Utility-based agent



Schematic diagram of a utility-based reflex agent

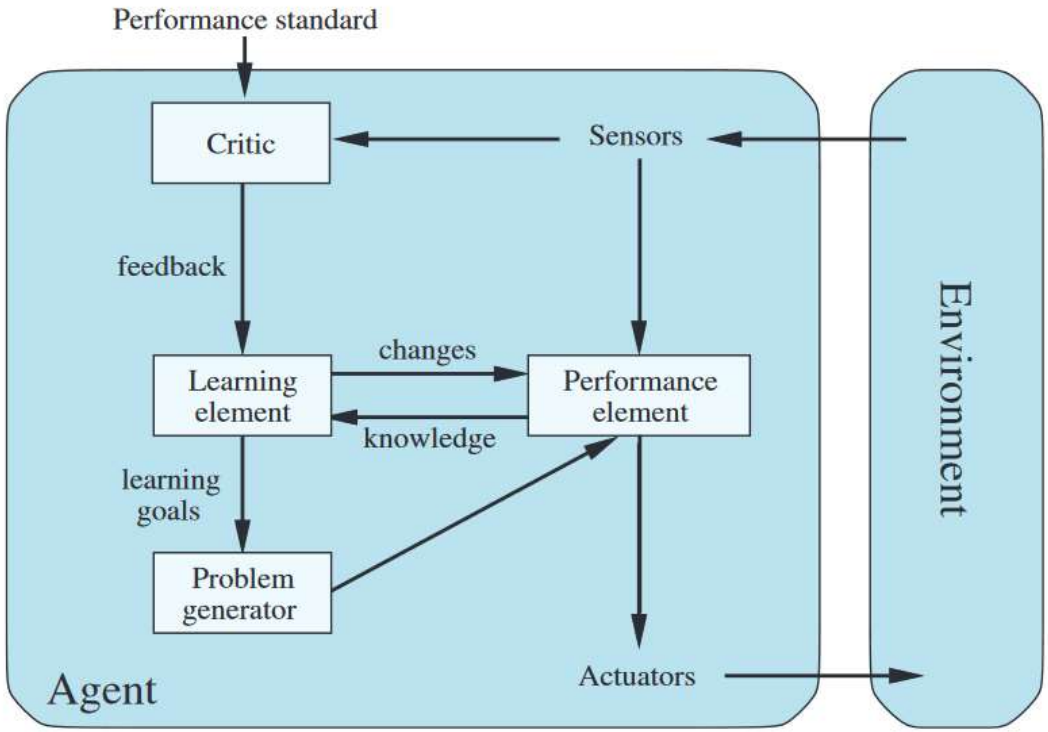
Learning agent

- After an agent is programmed, can it work immediately?
  - No, it still need teaching
- In AI, Once an agent is done
  - We teach it by giving it a set of examples
  - Test it by using another set of examples
- We then say the agent learns
  - A learning agent
- Learning has an advantage that it allows the agents to initially operate in unknown environments and to become more competent than its initial knowledge alone might allow.

Components of learning agent

- A learning agent can be divided into four conceptual components-
  - **Learning element-** responsible for making improvements.
  - **Performance elements-** responsible for selecting actions.
  - **Critics-** provides feedback
  - **Problem generator-** responsible for suggesting actions that will lead to new experiences.

Learning agent

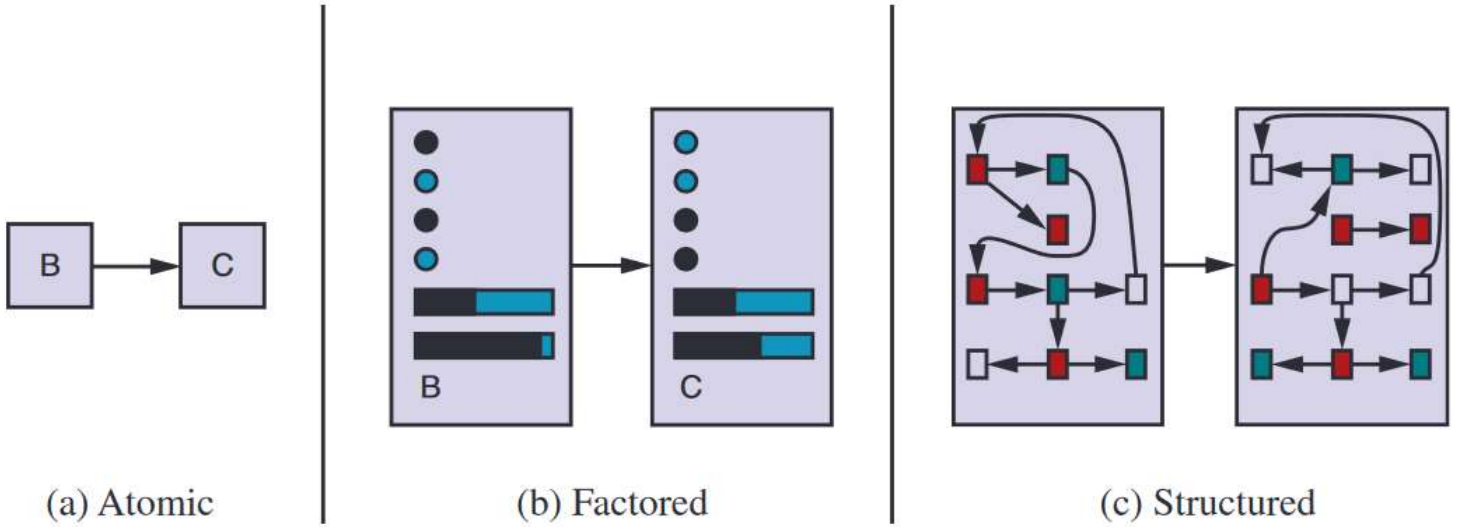


Schematic diagram of a learning agent

How the components of agent programs work?

- Agent programs (in very high-level terms) consist of various components, whose function it is to answer questions such as:
  - *"What is the world like now?"*
  - *"What action should I do now?"*
  - *"What do my actions do?"*
- The next question for a student of AI is, *"How on earth do these components work?"*
- Different ways to represent states and the transitions between them-
  - **Atomic**
  - **Factored**
  - **structured**

Ways to represent states and the transitions between them



How the components of agent programs work?

- In an **atomic representation** each state of the world is indivisible—it has no internal structure.
  - Consider the problem of finding a driving route from one end of a country to the other via some sequence of cities. For the purposes of solving this problem, it may suffice to reduce the state of world to just the name of the city we are in—a single atom of knowledge; a "black box" whose only discernible property is that of being identical to or different from another black box.
- **Areas of AI based on atomic representation are-**
  - Search and game-playing
  - Hidden Markov models
  - Markov decision processes etc.

How the components of agent programs work?

- A **factored representation** splits up each state into a fixed set of variables or attributes, each of which can have a value.
  - Two different atomic states have nothing in common—they are just different black boxes. But two different factored states can share some attributes (such as being at some particular GPS location) and not others (such as having lots of gas or having no gas); this makes it much easier to work out how to turn one state into another.
- With factored representations, we can also represent uncertainty—for example, ignorance about the amount of gas in the tank can be represented by leaving that attribute blank.
- **Areas of AI based on factored representation are-** ***constraint satisfaction algorithms, propositional logic, planning, Bayesian networks, machine learning algorithms etc.***

How the components of agent programs work?

- In **structured representation,** we need to understand the world as having things in it that are related to each other, not just variables with values.
  - For example, we might notice that a large truck ahead of us is reversing into the driveway of a dairy farm but a cow has got loose and is blocking the truck's path. A factored representation is unlikely to be pre-equipped with the attribute

    TruckAheadBackingIntoDairyFarmDrivewayBlockedByLooseCow

    with value true or false. Instead, we would need a structured representation, in which objects such as cows and trucks and their various and varying relationships can be described explicitly.
- **Areas of AI based on structured representation are-** ***relational databases and first-order logic, first-order probability models, knowledge-based learning and much of natural language understanding***

END OF CHAPTER 2