ARTIFICIAL INTELLIGENCE (A.I.)

Contents

What is an Agent?

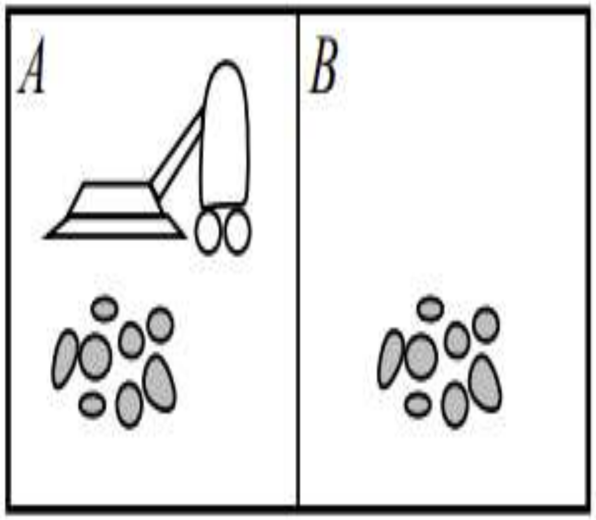
An agent can be anything that perceive its environment through sensors and act upon that environment through actuators. An Agent runs in the cycle of **perceiving**, **thinking**, and **acting**.

An agent can be:

- **Human-Agent:** A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract work for actuators.
- **Robotic Agent:** A robotic agent can have cameras, infrared range finder, NLP for sensors and various motors for actuators.
- **Software Agent:** Software agent can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen.

Hence the world around us is full of agents such as thermostat, cellphone, camera, and even we are also agents.

Example: Vacuum Cleaner world



Percepts: location and contents, e.g., [*A*, *Dirty*]

Actions: *Left*, *Right*, *Suck*, *NoOp*

Percept Sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
…..	

Intelligent Agents

An intelligent agent is an autonomous entity which act upon an environment using sensors and actuators for achieving goals. An intelligent agent may learn from the environment to achieve their goals.

Following are the main four rules for an AI agent:

- **Rule 1:** An AI agent must have the ability to perceive the environment.
- **Rule 2:** The observation must be used to make decisions.
- **Rule 3:** Decision should result in an action.
- **Rule 4:** The action taken by an AI agent must be a rational action.

Rational Agent

- A rational agent is an agent which has clear preference, models uncertainty, and acts in a way to maximize its performance measure with all possible actions.

- A rational agent is said to perform the right things. AI is about creating rational agents to use for game theory and decision theory for various real-world scenarios.

Omniscient Agents

An **omniscient agent** is an **agent** which knows the actual outcome of its action in advance. A chess **AI** can be a good example of a rational **agent** because, with the current action, it is not possible to foresee every possible outcome whereas a tic-tac-toe **AI** is **omniscient** as it always knows the outcome in advance.

Rationality differs from Omniscience because an Omniscient agent knows the actual outcome of its action and act accordingly, which is not possible in reality.

Task Environment

The "problems" for which rational agents are the "solutions" -- **PEAS (Performance measure, Environment, Actuator, Sensor)**

Performance Measure: Performance measure is the unit to define the success of an agent. Performance varies with agents based on their different percept.

Environment: Environment is the surrounding of an agent at every instant. It keeps changing with time if the agent is set in motion. There are 5 major types of environments:

- Fully Observable & Partially Observable
- Episodic & Non-Episodic
- Static & Dynamic
- Discrete & Continuous
- Deterministic & Stochastic

Actuator: Actuator is a part of the agent that delivers the output of an action to the environment.

Sensor: Sensors are the receptive parts of an agent which takes in the input for the agent.

Example of PEAS

Agent: self-driving cars

- **Performance measure:** Safety, time, legal drive, comfort
- **Environment:** Roads, other vehicles, road signs, pedestrian
- **Actuators:** Steering, accelerator, brake, signal, horn
- **Sensors:** Camera, GPS, speedometer, odometer, accelerometer, sonar.

Agent: Medical diagnosis system

- **Performance measure:** Healthy patient, minimize costs, lawsuits
- **Environment:** patient, hospital, staff
- **Actuators:** Screen display (question, tests, diagnoses, treatment, referrals)
- **Sensors:** Keywords (entry of symptoms, finding, patient's, answer)

Properties of Task Environment

The environment has multifold properties −

1. **Discrete / Continuous** − If there are a limited number of distinct, clearly defined, states of the environment, the environment is discrete (For example, chess); otherwise it is continuous (For example, driving).

2. **Observable / Partially Observable** − If it is possible to determine the complete state of the environment at each time point from the percepts it is observable; otherwise it is only partially observable.

3. **Static / Dynamic** − If the environment does not change while an agent is acting, then it is static; otherwise it is dynamic.

4. **Single agent / Multiple agents** − The environment may contain other agents which may be of the same or different kind as that of the agent.

5. **Accessible / Inaccessible** − If the agent’s can have access to the complete state of the environment, then the environment is accessible to that agent.

6. **Deterministic / Non-deterministic** − If the next state of the environment is completely determined by the current state and the actions of the agent, then the environment is deterministic; otherwise it is non-deterministic.

7. **Episodic / Non-episodic** − In an episodic environment, each episode consists of the agent perceiving and then acting. The quality of its action depends just on the episode itself. Subsequent episodes do not depend on the actions in the previous episodes. Episodic environments are much simpler because the agent does not need to think ahead.

Structure of Intelligent Agents

The task of AI is to design an **agent program** which implements the agent function(mapping of percepts to actions). We assume this program will run on some sort of computing device which we call the **architecture**.

The **architecture might be** a plain computer or special purpose hardware for certain tasks, such as processing camera images or filtering audio input. It might also include software that provide a degree of insulation between the raw computer and the agent program.

The structure of an intelligent agent is a combination of architecture and agent program. It can be viewed as:

Agent = Architecture + Agent program

The appropriate design of the **agent** program depends on the **nature** of the environment. Architecture is a computing device used to run the **agent** program.

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

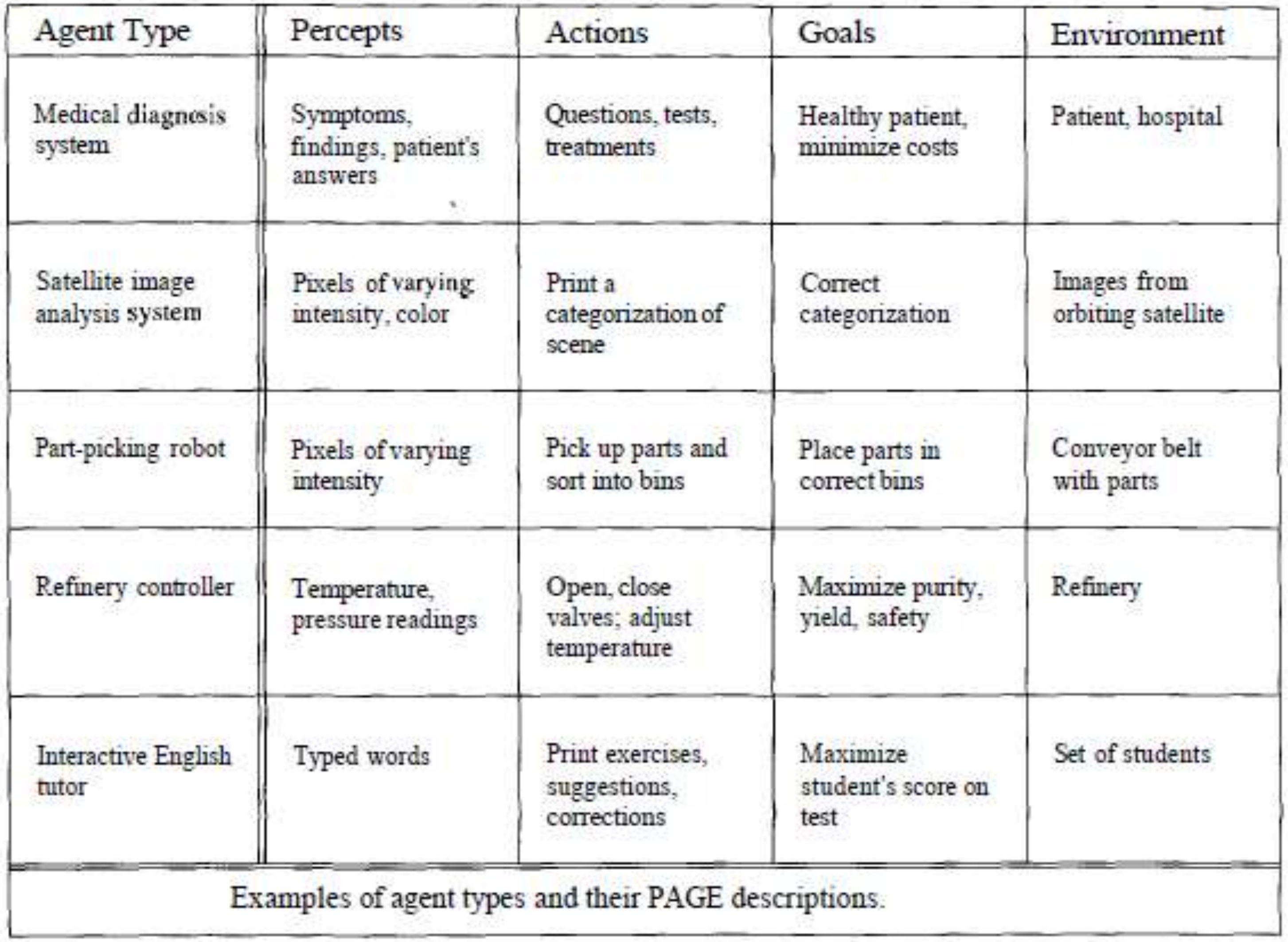
Architecture: Architecture is machinery that an AI agent executes on.

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

$$\mathbf{f:P^* \rightarrow A}$$

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

Agent Types

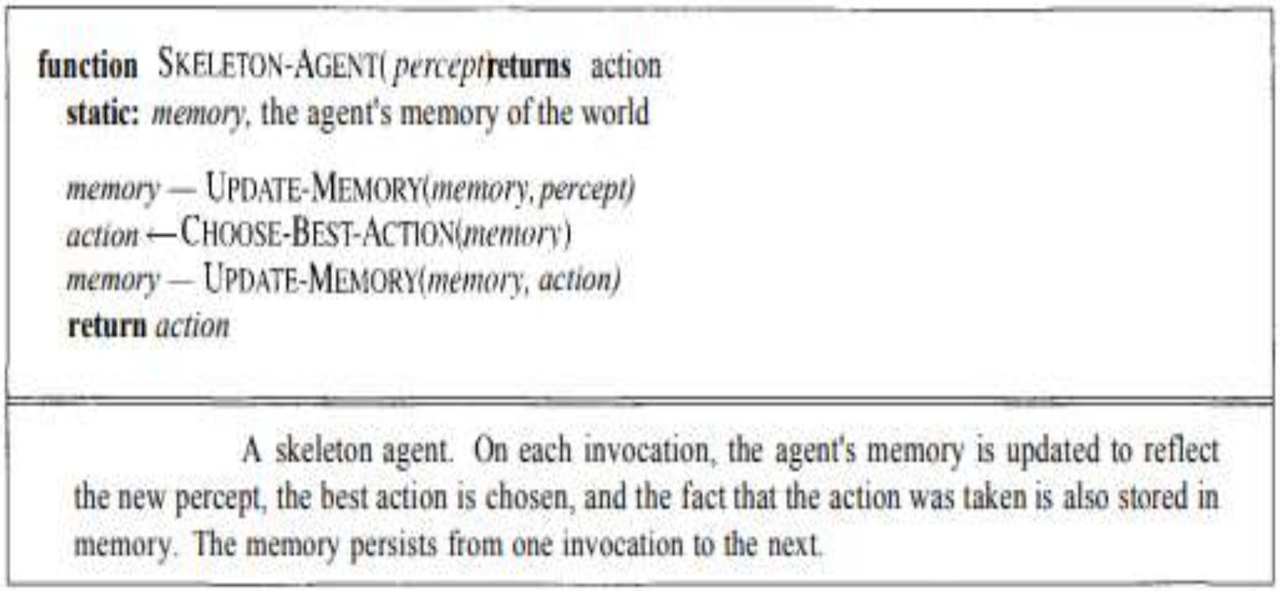


Agent Type	Percepts	Actions	Goals	Environment
Medical diagnosis system	Symptoms, findings, patient's answers	Questions, tests, treatments	Healthy patient, minimize costs	Patient, hospital
Satellite image analysis system	Pixels of varying intensity, color	Print a categorization of scene	Correct categorization	Images from orbiting satellite
Part-picking robot	Pixels of varying intensity	Pick up parts and sort into bins	Place parts in correct bins	Conveyor belt with parts
Refinery controller	Temperature, pressure readings	Open, close valves; adjust temperature	Maximize purity, yield, safety	Refinery
Interactive English tutor	Typed words	Print exercises, suggestions, corrections	Maximize student's score on test	Set of students

Examples of agent types and their PAGE descriptions.

Agent Program

Agent programs have a very simple form. Each will use some internal data structures that will be updated as new percepts arrive. These 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 be executed.



```
function SKELETON-AGENT(percept) returns action
  static: memory, the agent's memory of the world

  memory ← UPDATE-MEMORY(memory, percept)
  action ← CHOOSE-BEST-ACTION(memory)
  memory ← UPDATE-MEMORY(memory, action)
  return action
```

A skeleton agent. On each invocation, the agent's memory is updated to reflect the new percept, the best action is chosen, and the fact that the action was taken is also stored in memory. The memory persists from one invocation to the next.

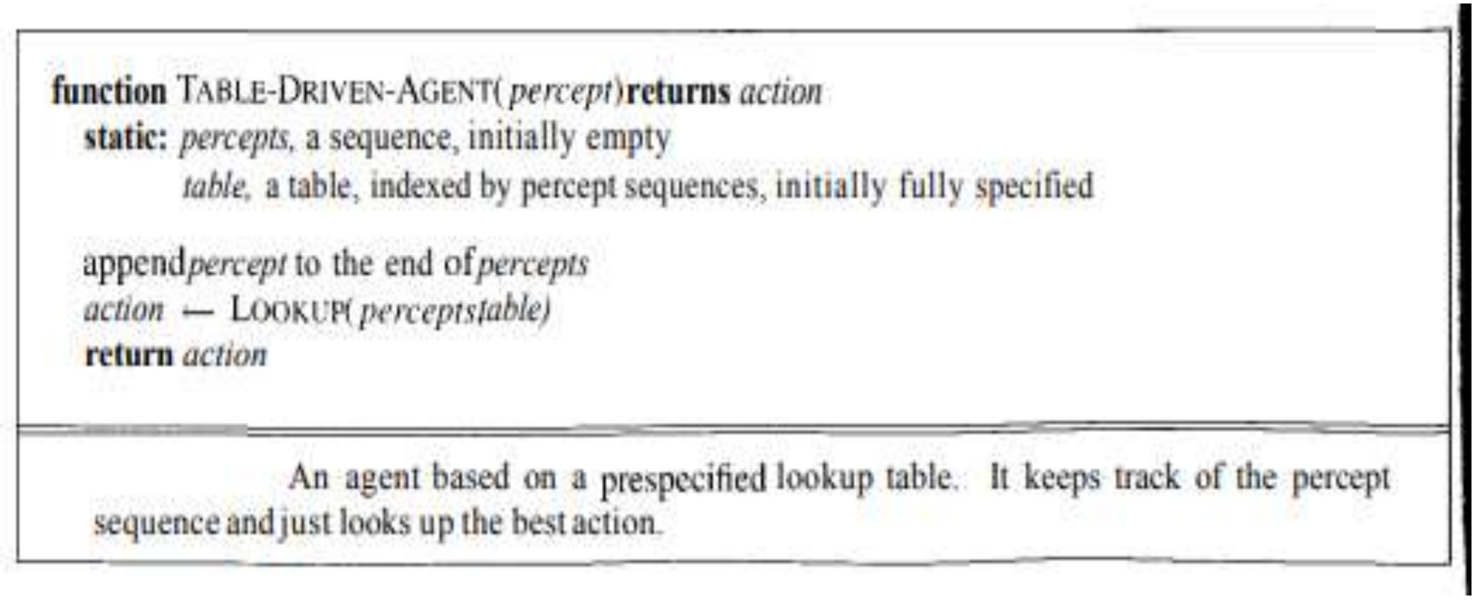
There are **two things** about this skeleton program.

First, even though we defined the agent mapping as a function from percept sequences to actions, the agent program receives only a single percept as its input. It is up to the agent to build up the percept sequence in memory, if it so desires. In some environments, it is possible to be quite successful without storing the percept sequence, and in complex domains, it is infeasible to store the complete sequence.

Second, the goal or performance measure is not part of the skeleton program. This is because the performance measure is applied externally to judge the behavior of the agent.

Why not just look up the answers?

The simplest possible way is that we can write the agent program—a lookup table. It operates by keeping in memory its entire percept sequence, and using it to index into table, which contains the appropriate action for all possible percept sequences.



```
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
```

An agent based on a prespecified lookup table. It keeps track of the percept sequence and just looks up the best action.

Types of Agent Program

Agents can be grouped into five classes based on their degree of perceived intelligence and capability. All these agents can improve their performance and generate better action over the time. These are

- Simple Reflex Agent
- Model-based reflex agent
- Goal-based agents
- Utility-based agent
- Learning agent

Simple Reflex agent

- The Simple reflex agents are the simplest agents. These agents take decisions on the basis of the current percepts and ignore the rest of the percept history.

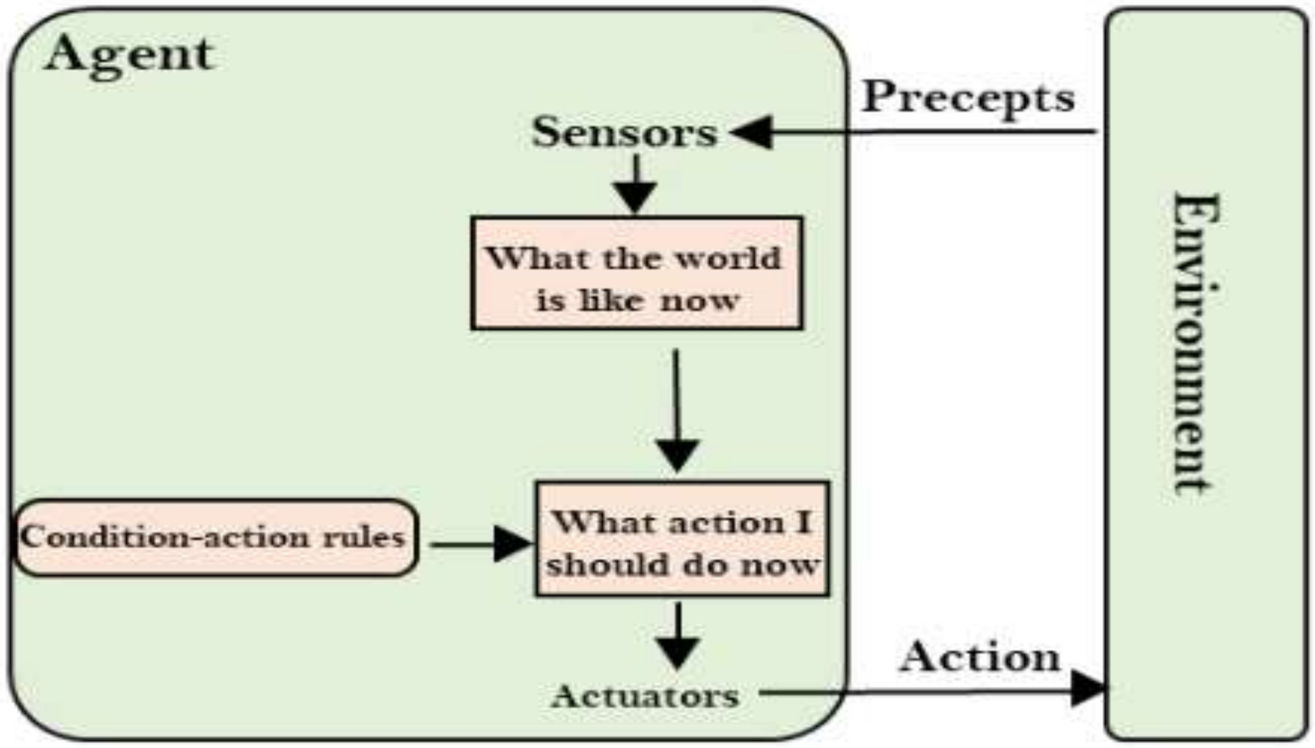
- These agents only succeed in the **fully observable environment**.

- The Simple reflex agent does not consider any part of percepts history during their decision and action process.

- The Simple reflex agent works on Condition-action rule, which means it maps the current state to action.

Problems for the simple reflex agent design approach:

- They have very limited intelligence
- Not adaptive to changes in the environment.

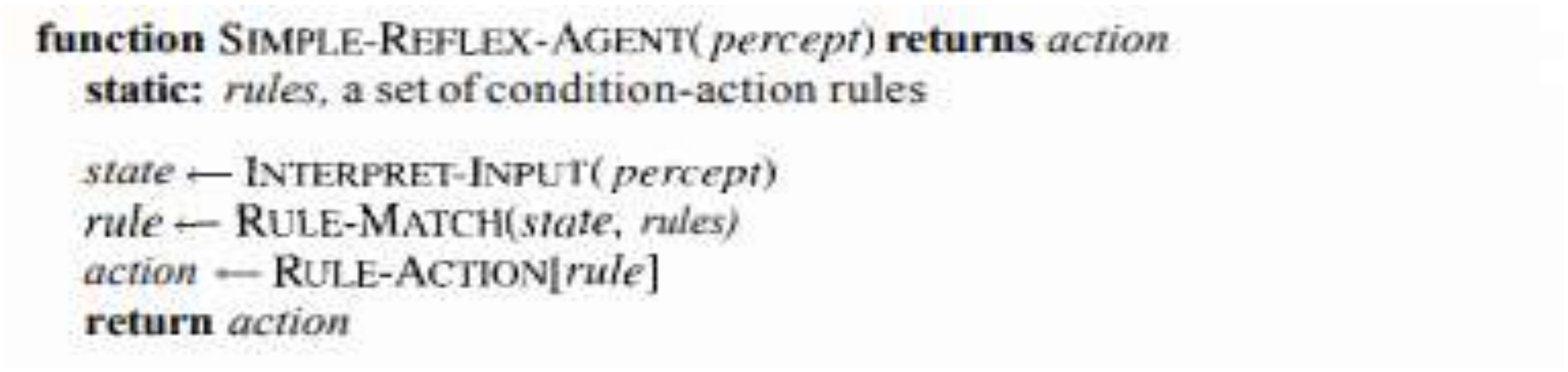


Example:

The vacuum promises to sense dirt and debris(scattered pieces) on your floors and clean those areas accordingly. This is an **example** of a **simple reflex agent** that operates on the condition (dirty floors) to initiate an action (vacuuming).

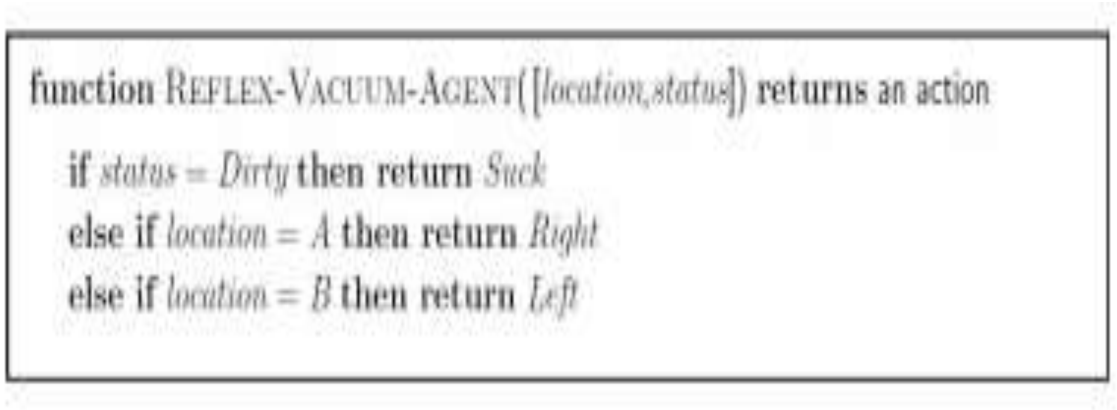
AGENT PROGRAM OF SIMPLE REFLEX AGENT:

The agent program is very simple and shown . The INTERPRET-INPUT function generates an abstracted description of the current state from the percept, and the RULE-MATCH function returns the first rule in the set of rules that matches the given state description. Although such agents can be implemented very efficiently , their range of applicability is very narrow.



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

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



```
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
```

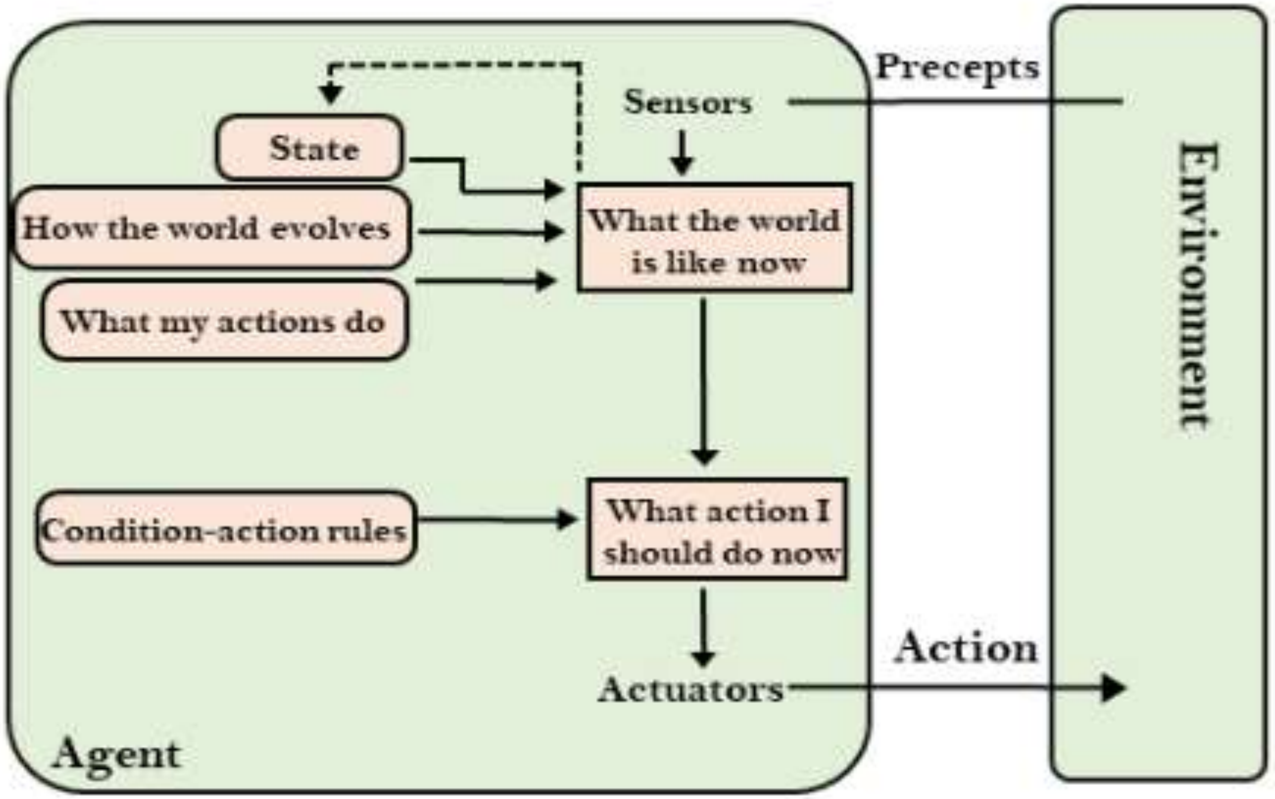
Model-based reflex agent

- The Model-based agent can work in a **partially observable environment,** and track the situation.

- A model-based agent has two important factors:
  - **Model:** It is knowledge about "how things happen in the world," so it is called a Model-based agent.
  - **Internal State:** It is a representation of the current state based on percept history.

- These agents have the model, "which is knowledge of the world" and based on the model they perform actions.

- Updating the agent state requires information about:
  - How the world evolves
  - How the agent's action affects the world.



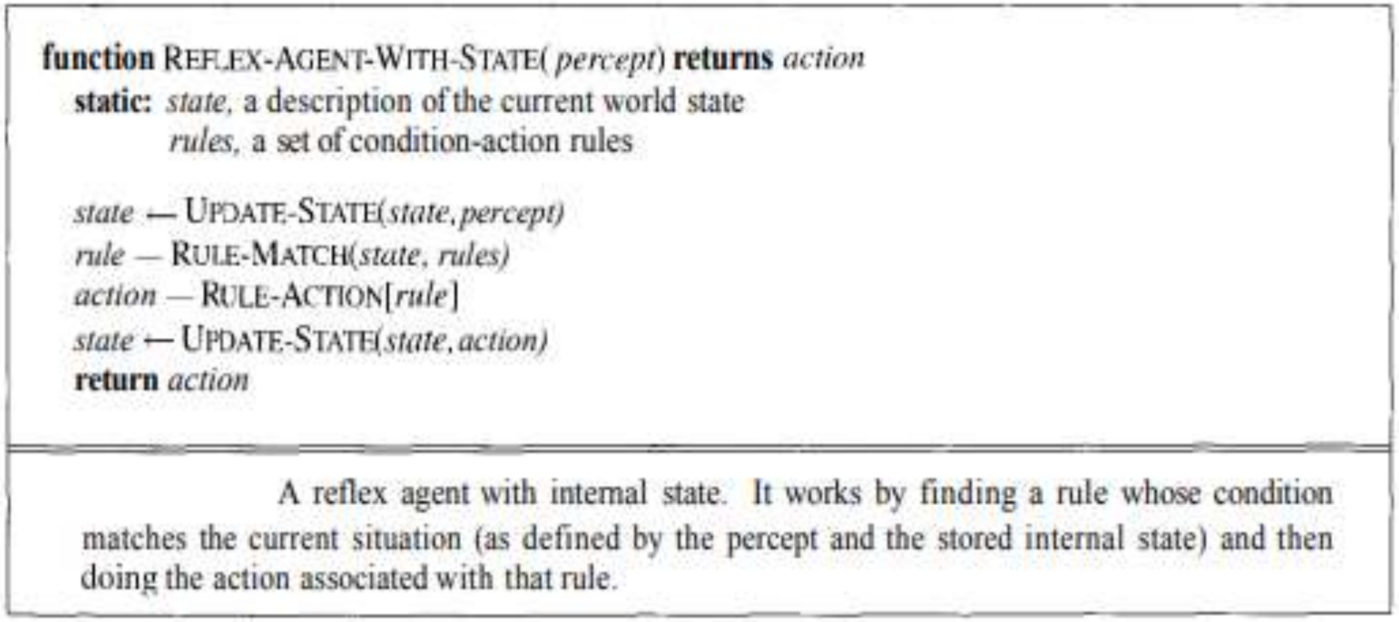
Example: Waymo

- Google's started autonomous car project in 2009 known as Waymo, and it is in a public trial phase in Phoenix, Arizona. It's also representative of an **intelligent agent**, an entity that can observe its environment through the use of sensors and then take action using actuators.

- Waymo is a specific type of intelligent agent known as a model-based agent.

AGENT PROGRAM OF MODEL BASED AGENT:

A reflex agent with internal state. It works by finding a rule whose condition matches the current situation (as defined by the percept and the stored internal state) and then doing the action associated with that rule.



```
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
```

A reflex agent with internal state. It works by finding a rule whose condition matches the current situation (as defined by the percept and the stored internal state) and then doing the action associated with that rule.

Goal-based agents

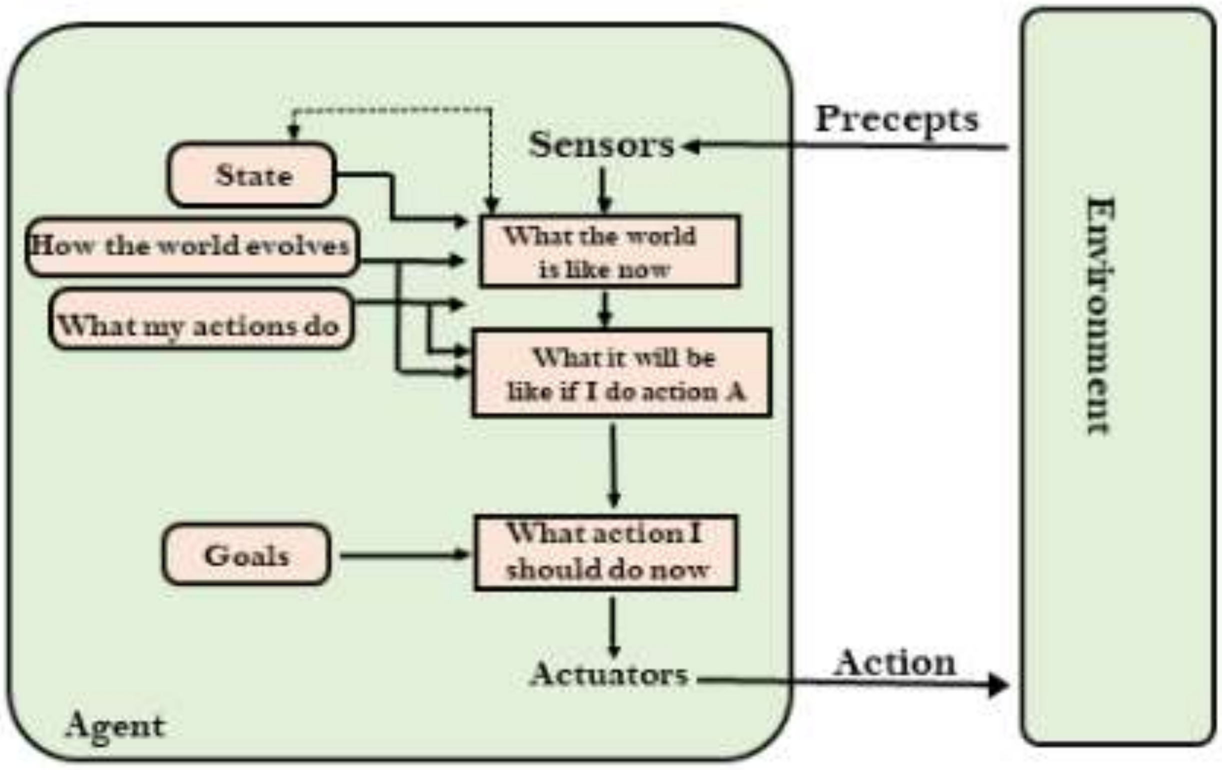
- The knowledge of the current state environment is not always sufficient to decide for an agent to what to do.

- The agent needs to know its goal which describes desirable situations.

- **Goal-based agents expand the capabilities of the model-based agent by having the "goal" information.**

- They choose an action, so that they can achieve the goal.

- These agents may have to consider a long sequence of possible actions before deciding whether the goal is achieved or not. Such considerations of different scenario are called **searching and planning**, which makes an agent **proactive(perform actions based on prediction).**



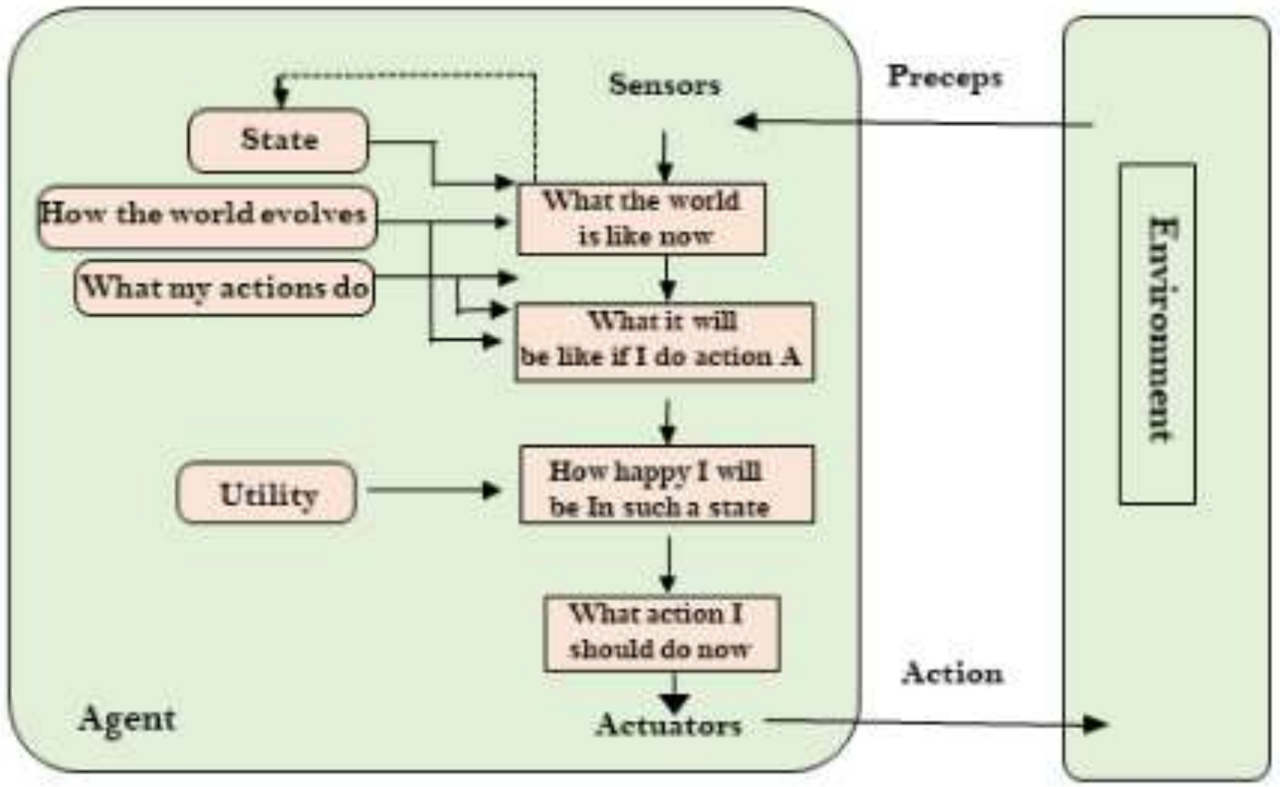
Example:

Google's Waymo driverless cars are good **examples** of a **goal-based agent** when they are programmed with an end destination, or **goal**, in mind.

The car will then "think" and make the right decisions in order to deliver the passenger where they intended to go.

Utility-based agents

- **These agents are similar to the goal-based agent but provide an extra component of utility measurement** which makes them different by providing a measure of success at a given state.
- **Utility-based agent act based not only on the goals but also the best way to achieve the goal.**
- The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action.
- The utility function maps each state to a real number to check how efficiently each action achieves the goals.


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

Example:

Home thermostat that knows to start heating or cooling your house **based** on reaching a certain temperature.

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.

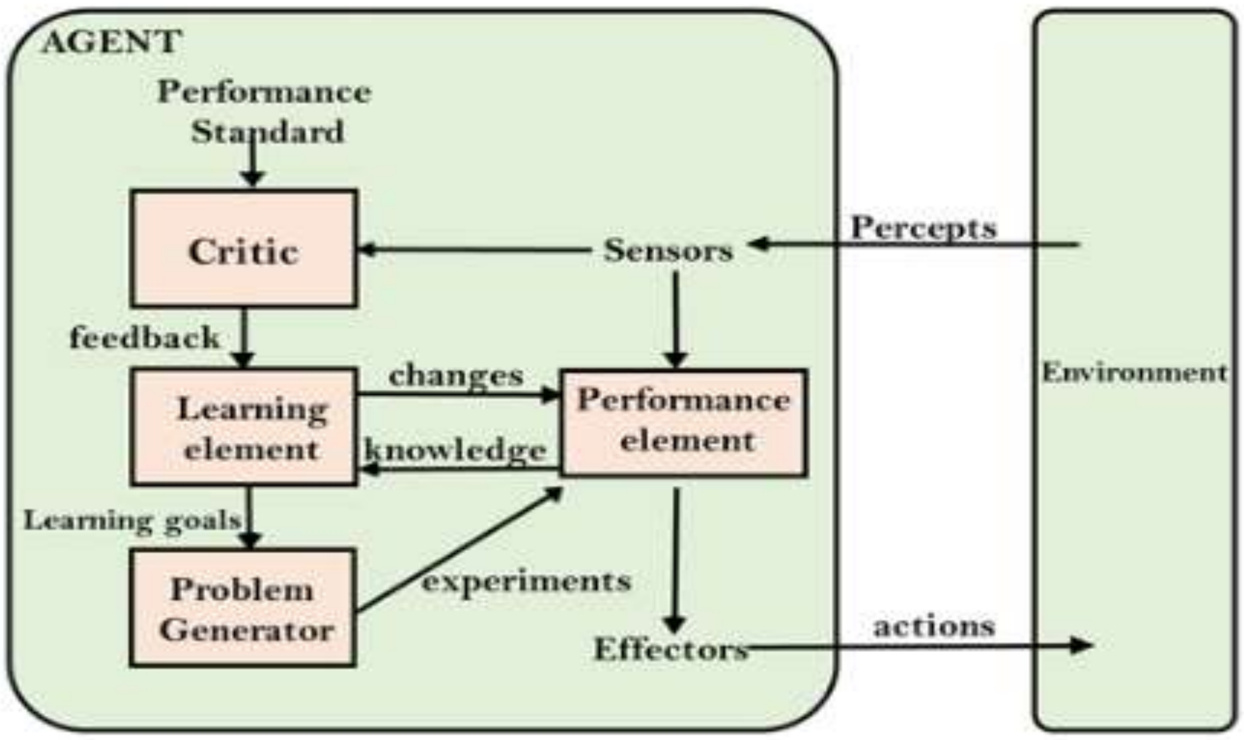
Learning Agents

- A learning agent in AI is the type of agent which can learn from its past experiences, or it has learning capabilities.

- It starts to act with basic knowledge and then able to act and adapt automatically through learning.

- A learning agent has mainly four conceptual components, which are:
  1. **Critic:** Learning element takes feedback from critic which describes that how well the agent is doing with respect to a fixed performance standard.
  2. **Learning element:** It is responsible for making improvements by learning from environment
  3. **Performance element:** It is responsible for selecting external action
  4. **Problem generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.

- Hence, learning agents are able to learn, analyze performance, and look for new ways to improve the performance.



Example:

When you were in school you would do a test and it would be marked the **test is the critic.**

The teacher would mark the test and see what could be improved and instructs you how to do better next time, the **teacher is the learning element** and **you are the performance element** responsible for selecting external action.

Communication among agents

Agent communication is based on message passing, where **agents communicate** by formulating and sending individual messages to each other.

Communication can be:

Point to Point- An agent talks directly to another agent.

Broadcast- An agent sends some information to a group of agents .

Mediated- The communication between two agents is mediated by a third party.

Multi-Agent system (MAS)

A system where multiple single agents work together is referred to as **multiagent system (MAS).**

MAS consists of multiple agents which may have common goal where they work together to achieve a certain task;

or have self-interested goal where each agent competes against each other for resources;

or they are required to coordinate to achieve a certain task.

Benefits of Multi Agent System

1. **Modularity:** Each agent is specialized in the solution of a particular kind of problems (leading also to reusability). The complexity of the construction of agents is reduced. The process of solving a complex problem is reduced to solving easier sub problems .

2. **Efficiency:** Problems can be solved more quickly, due to the inherent concurrency/parallelism .Different agents are working at the same time in different parts of a problem.

3. **Reliability:** Avoid single point of failure in centralized systems. If an individual agent fails, the other agents can take its work and redistribute it dynamically.

4. **Flexibility:** Agents can be created/deleted dynamically, depending on the amount of work to be done, the available resources, etc. Agents can dynamically generate subtasks and look for helping agents.

Why do we need Agent Communication?

Communication is necessary in order to allow collaboration(enables individuals to work together to achieve a defined and common purpose), negotiation, cooperation, coordination etc. between independent entities.

For this purpose, **a common language protocol** is required in order to achieve interoperability(the ability of computer systems to exchange and make use of information).

These interactions can only be carried out if the agents can communicate and understand the communicated message syntactically and semantically.

Thus, **agent communication language** has been developed in order for agent to communicate with each other and understand the content of communication.

Agent Communication Language

- Agent communication language (ACL) is a vital component in multiagent system (MAS) to enable the agents to communicate and exchange messages and knowledge.

- ACL allows more complex knowledge exchange such as plans, agent's goal, and believes which cannot be exchanged using object-oriented approach.

- Different agent communication languages and different semantic models have been developed to ease the communication between agents in MAS.

- Agent Communication Language (ACL), proposed by the Foundation for Intelligent Physical Agents (FIPA), is a proposed standard language for agent communications. Knowledge Query and Manipulation Language (KQML) is another proposed standard.

- **The most popular ACLs are:**

  1. **FIPA-ACL** (by the Foundation for Intelligent Physical Agents, a standardization consortium)
  2. **KQML** (Knowledge Query and Manipulation Language)

KQML

The Knowledge Query and Manipulation Language (KQML) is a language and protocol for exchanging information and knowledge.

KQML is a high level, message-oriented communication language and protocol for information exchange with an intelligent system by an application program or another intelligent system.

KQML Layers

There are **three layers** involved in KQML:

1. The content Layer
2. The message Layer
3. The communication Layer

The Content Layer: contains the actual content of the message.

The Communication Layer: describes the lower level communication parameters such as the identity of the sender and the receiver and the unique identifier associated with the communication.

The Message Layer: forms the core of KQML. It encodes the message that one application would like to transmit to another. The layer determines the kinds of interactions one agent may have with another.

KQML Reserved Parameter Keywords

The KQML format comprises number of fields such as the sender of the message, the list of receivers, the communication primitives.

Communication primitives can be

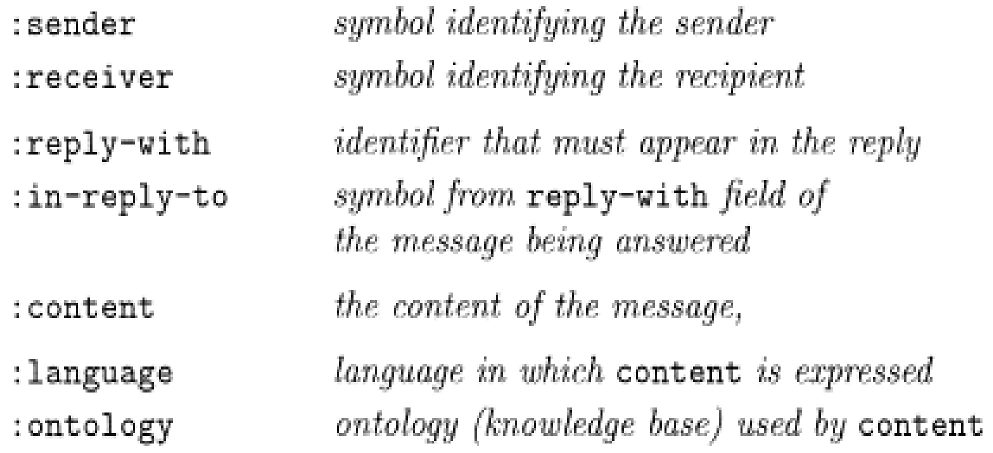
request-if : the sender wants the receiver to perform an action.

inform-if: the sender wants the receiver to be aware of the fact.

query-if: if the sender wants to know whether or not a given condition holds.

Proposal:--

accept _proposal ; reject_proposal : if the sender and receiver are engaged in a negotiation and more.



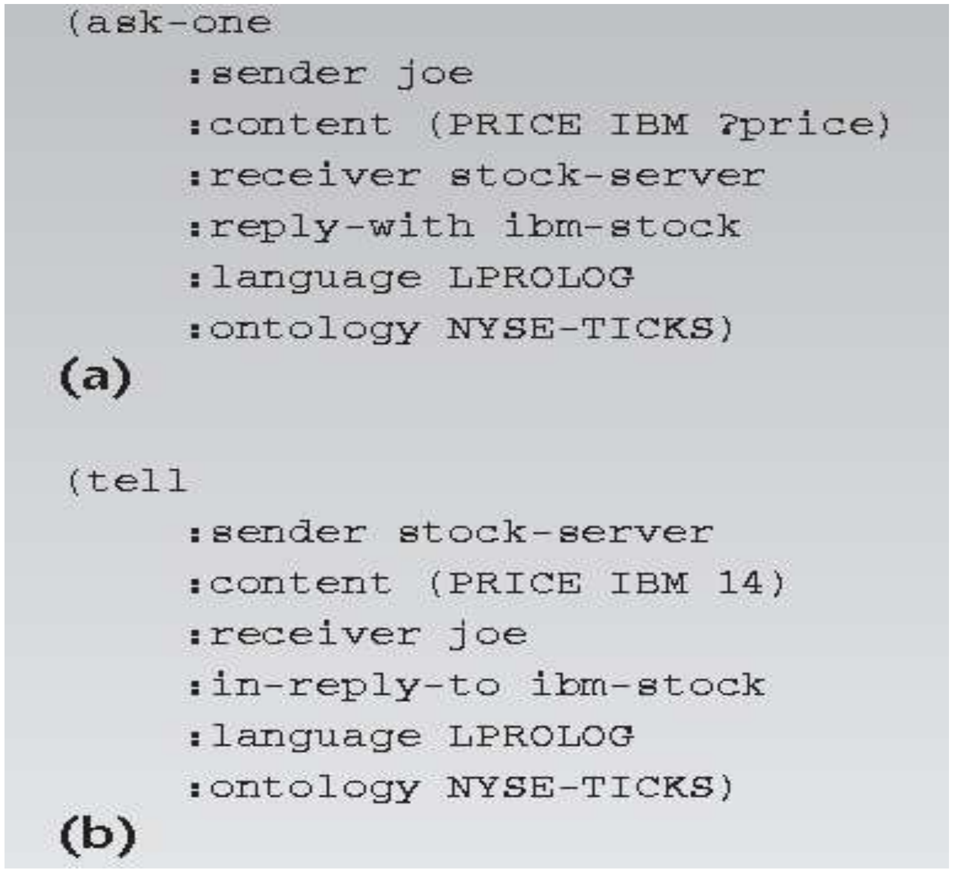
`:sender`	*symbol identifying the sender*
`:receiver`	*symbol identifying the recipient*
`:reply-with`	*identifier that must appear in the reply*
`:in-reply-to`	*symbol from* `reply-with` *field of the message being answered*
`:content`	*the content of the message,*
`:language`	*language in which* `content` *is expressed*
`:ontology`	*ontology (knowledge base) used by* `content`

Example of KQML

Examples of messages in KQML:

(a) a query from agent joe about the price of IBM stock

(b) the stock-server's reply.



```
(ask-one
      :sender joe
      :content (PRICE IBM ?price)
      :receiver stock-server
      :reply-with ibm-stock
      :language LPROLOG
      :ontology NYSE-TICKS)
```

(a)

```
(tell
      :sender stock-server
      :content (PRICE IBM 14)
      :receiver joe
      :in-reply-to ibm-stock
      :language LPROLOG
      :ontology NYSE-TICKS)
```

(b)

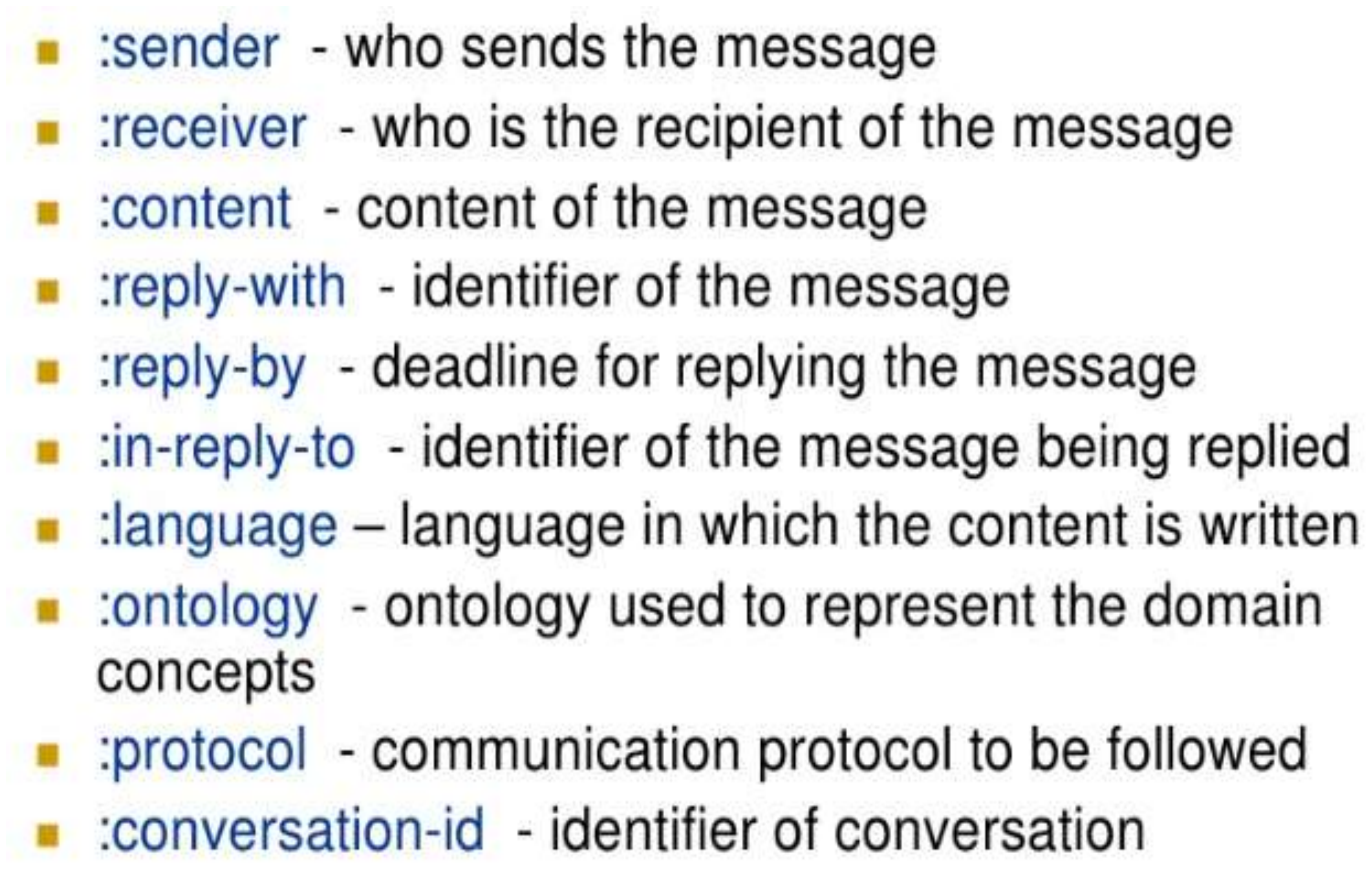
FIPA-ACL

The FIPA is acronym for **‘Foundation for Intelligent Physical Agents’.**

The FIPA has defined agent communication language for agent inter-operability and specified format for messages exchanged by the agents.

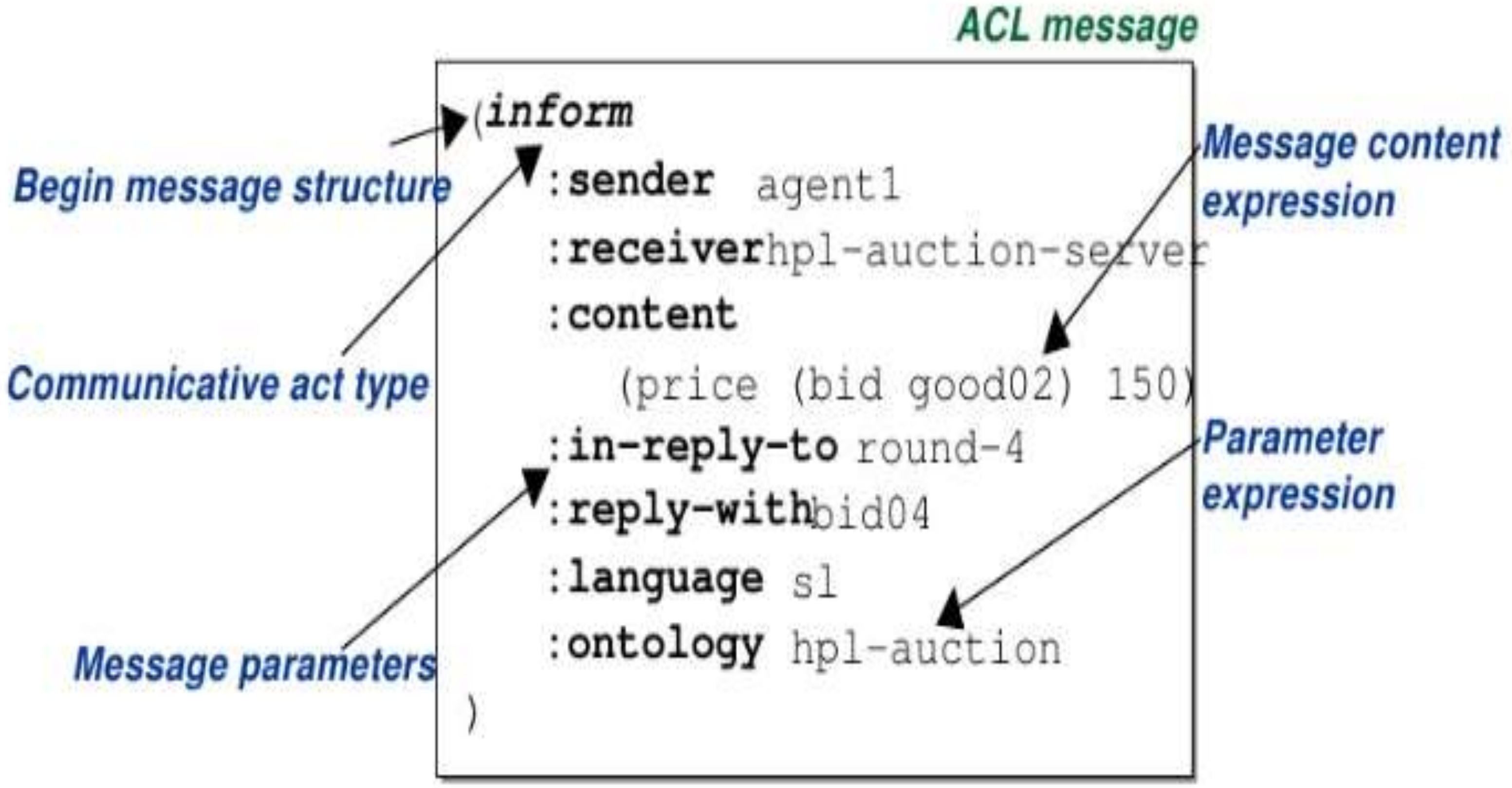
Its syntax is similar to that of KQML except for different names for some reserved primitives.

Parameters in a FIPA ACL message



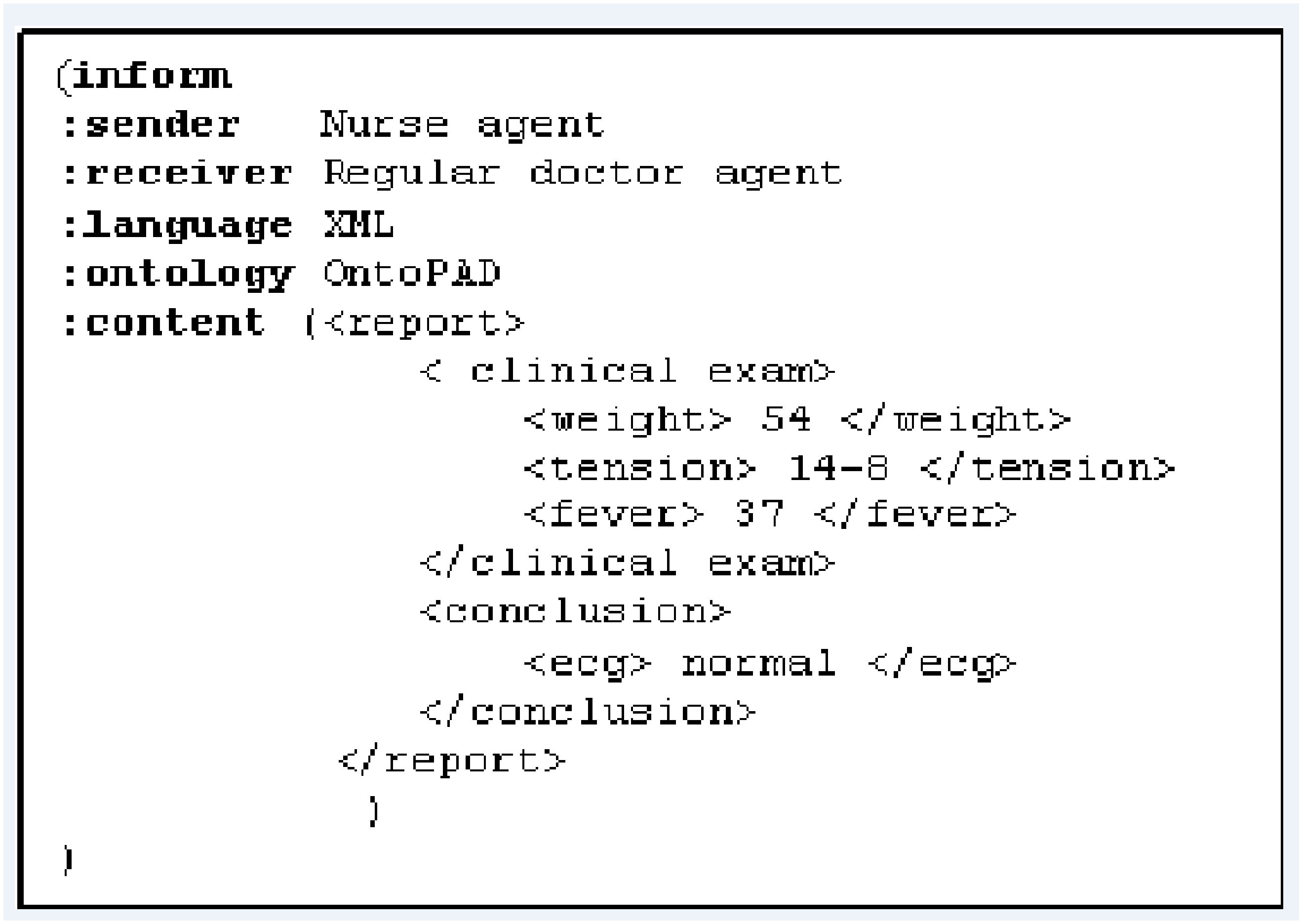
- :sender - who sends the message
- :receiver - who is the recipient of the message
- :content - content of the message
- :reply-with - identifier of the message
- :reply-by - deadline for replying the message
- :in-reply-to - identifier of the message being replied
- :language – language in which the content is written
- :ontology - ontology used to represent the domain concepts
- :protocol - communication protocol to be followed
- :conversation-id - identifier of conversation

Syntax of FIPL ACL



```
(inform
  :sender  agent1
  :receiver hpl-auction-server
  :content
     (price (bid good02) 150)
  :in-reply-to round-4
  :reply-with bid04
  :language sl
  :ontology hpl-auction
)
```

Example of FIPA-ACL



```
(inform
:sender   Nurse agent
:receiver Regular doctor agent
:language XML
:ontology OntoPAD
:content (<report>
              < clinical exam>
                   <weight> 54 </weight>
                   <tension> 14-8 </tension>
                   <fever> 37 </fever>
              </clinical exam>
              <conclusion>
                   <ecg> normal </ecg>
              </conclusion>
          </report>
           )
)
```