Introduction to Intelligent Agent

By Dr. Gopal Chandra Jana, Assistant Professor, DCSE, SSCSE

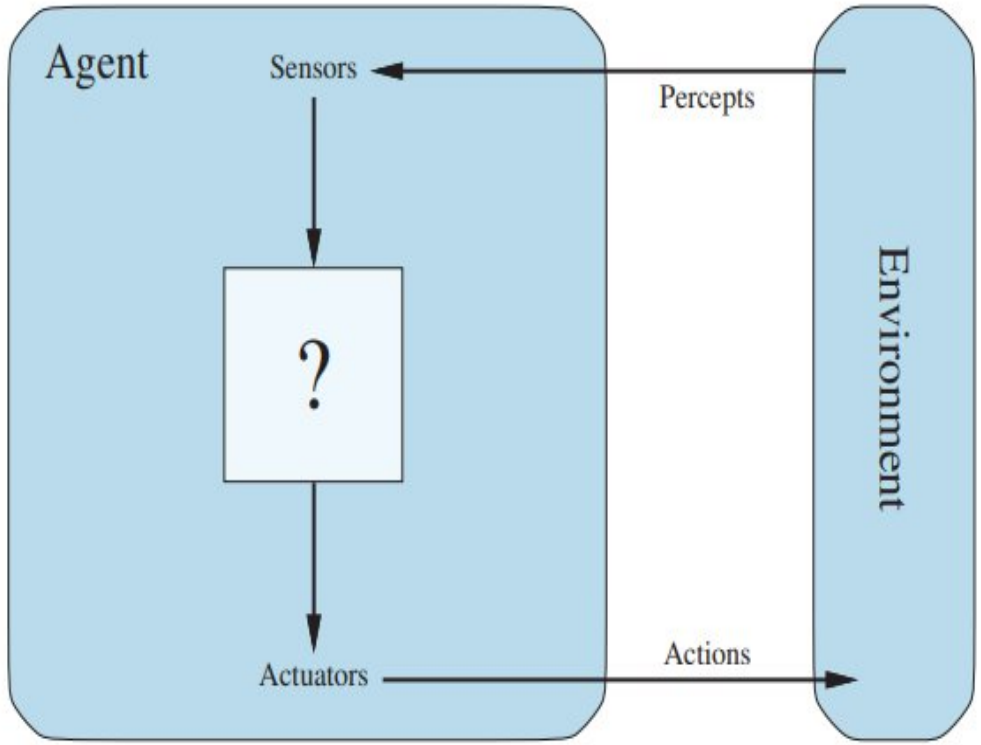
Course Page:
https://www.gcjana.in/courses/shardauniversity/2501/CSE472/

Topic to be covered

1. Agents, Environments, Structure of Agent
2. PEAS (Performance, Environment, Actuators, Sensors)
3. PAGE (Percepts, Actions, Goals, Environment)
4. Rationality and Rational Agent
5. Types of Agents
6. Agent Program

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

Human Agent	Robotic Agent	Software Agent
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 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.

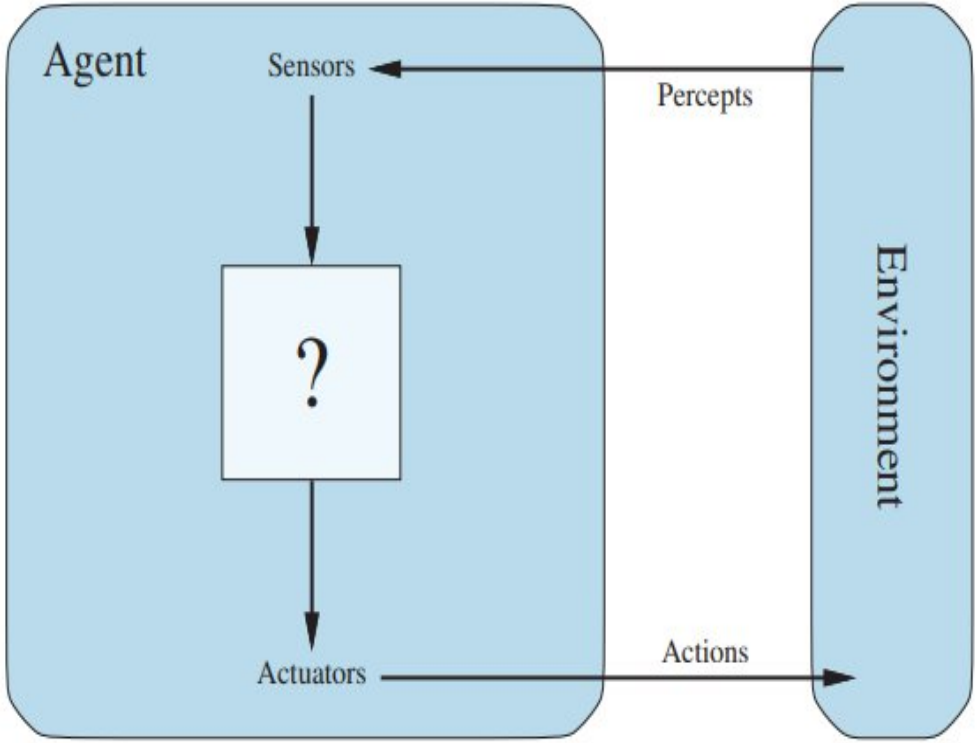
Example: What is the driving environment that the taxi will face? Any taxi driver must deal with a variety of roads, ranging from rural lanes and urban alleys to 12-lane freeways. The roads contain other traffic, pedestrians, stray animals, road works, police cars, puddles, and potholes.

Sensors: Detect inputs from the environment (e.g., camera, temperature sensor).

Actuators: Carry out actions in the environment based on the agent's decisions.

The **actuators for an automated taxi** include those available to a human driver: control over the engine through the accelerator and control over steering and braking. In addition, it will need output to a display screen or voice synthesizer to talk back to the passengers, and perhaps some way to communicate with other vehicles, politely or otherwise.

The basic **sensors for the taxi** will include one or more video cameras so that it can see, as well as lidar and ultrasound sensors to detect distances to other cars and obstacles.



PAGE (Percepts, Actions, Goals, Environment)

PAGE

Must first specify the setting for intelligent agent design

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

Percepts??

Actions??

Goals??

Environment??

PAGE

Must first specify the setting for intelligent agent design

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

Percepts?? video, accelerometers, gauges, engine sensors, keyboard, GPS, ...

Actions?? steer, accelerate, brake, horn, speak/display, ...

Goals?? safety, reach destination, maximize profits, obey laws, passenger comfort, ...

Environment?? US urban streets, freeways, traffic, pedestrians, weather, customers, ...

Internet shopping agent

Percepts??

Actions??

Goals??

Environment??

For an **Internet Shopping Agent**, PAGE

Percepts:

- Product details (price, specifications, images, availability)
- User preferences (budget, brand, ratings, past purchases)
- Reviews and ratings
- Discounts, offers, delivery time, seller reputation
- Stock status and shipping costs

Actions:

- Search for products
- Filter and sort items (by price, rating, delivery time, etc.)
- Compare products across platforms
- Add/remove items from cart
- Place order / recommend best option
- Track order or notify user of price drops

Goals:

- Find the best product matching user requirements
- Minimize cost while maximizing quality and reliability
- Ensure timely delivery
- Improve user satisfaction and decision-making
- Save user time and effort

Environment:

- E-commerce websites and marketplaces (Amazon, Flipkart, etc.)
- Payment gatewaysUser device/browser
- Internet/network conditions
- Sellers, logistics, and review systems

PEAS (Performance, Environment, Actuators, Sensors)

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

PAGE Vs PEAS

For an Internet Shopping Agent, PAGE

P: Product details, prices, reviews, offers

A: Search products, compare, purchase

G: Buy best product at lowest price

E: E-commerce platforms and sellers

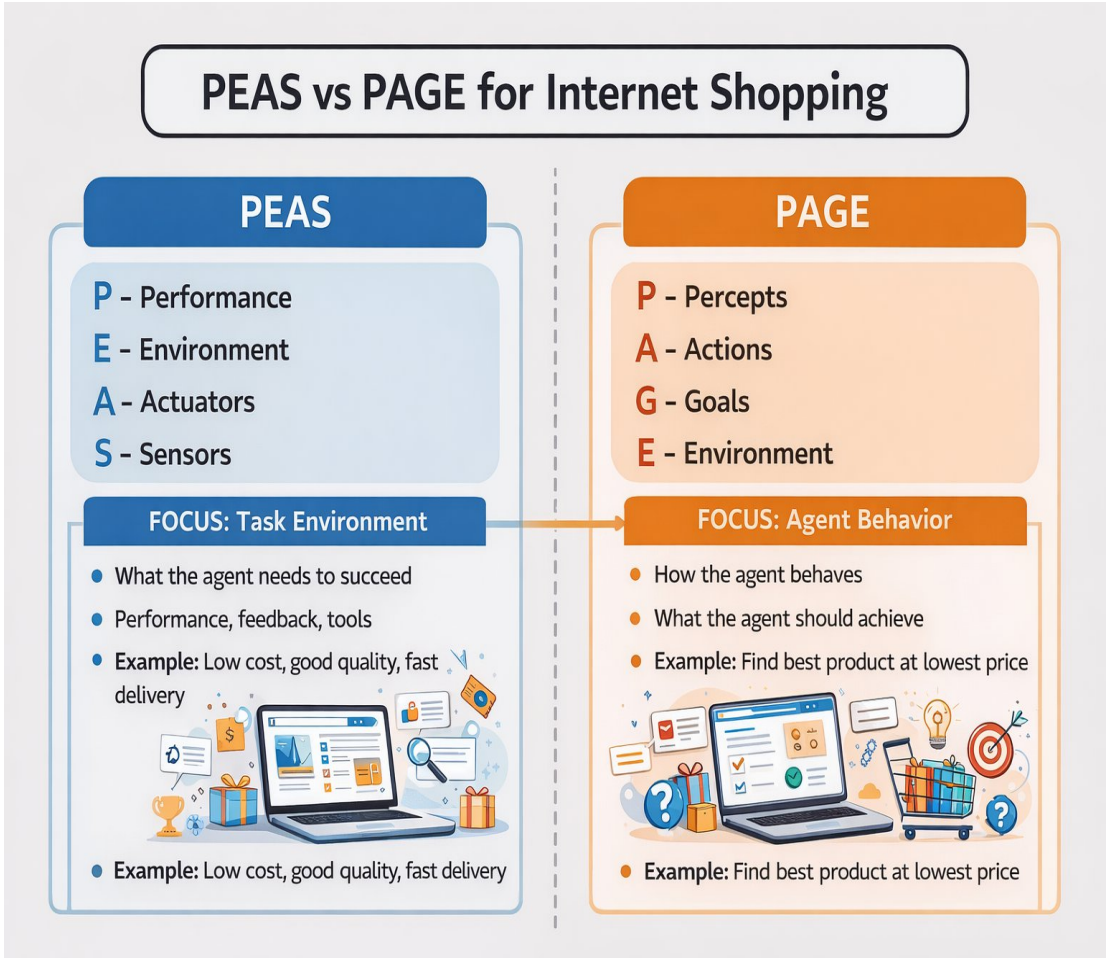
Internet Shopping Agent, PEAS

P: Low cost, good quality, timely delivery, user satisfaction

E: Online shopping websites, sellers, payment systems

A: Search, compare, add to cart, place order

S: Prices, reviews, availability, user preferences



Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments	Touchscreen/voice entry of symptoms and findings
Satellite image analysis system	Correct categorization of objects, terrain	Orbiting satellite, downlink, weather	Display of scene categorization	High-resolution digital camera
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, tactile and joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, raw materials, operators	Valves, pumps, heaters, stirrers, displays	Temperature, pressure, flow, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, feedback, speech	Keyboard entry, voice

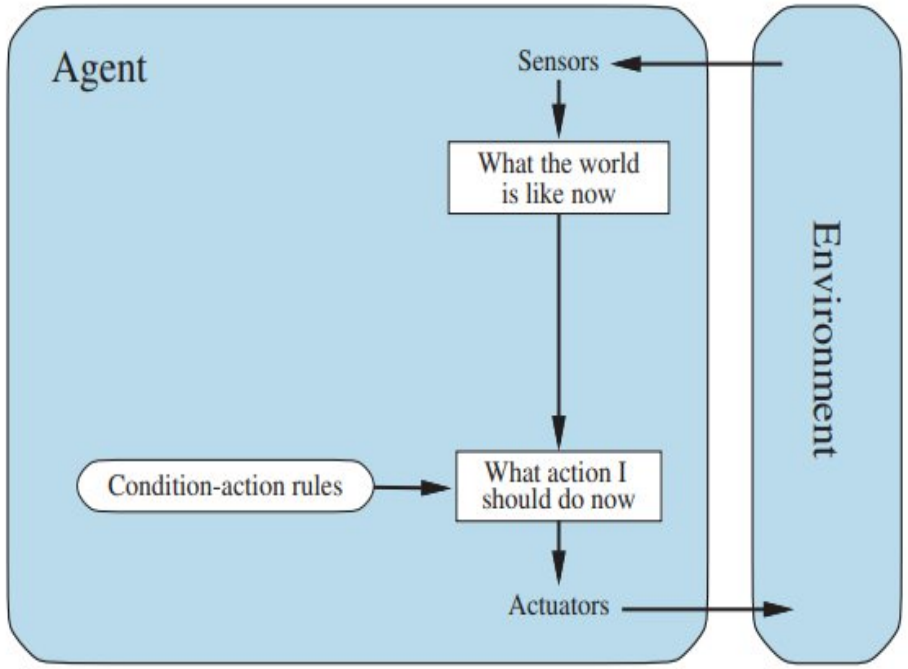
Agent types

Four basic types in order of increasing generality:

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

Simple reflex agent:

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.



About Simple reflex Agent: These agents act solely based on the current percept, ignoring the rest of the percept history.

Working Principle: They use condition – action rules: *"if condition, then action".*

Limitation: Cannot deal with partial observability or changes in the environment that require memory.

Example: A vacuum cleaner that turns right when it sees dirt.

Simple reflex agent:

Key Points

- Act **only on the current percept**
- Follow **IF–THEN rules**
- **No memory** of past actions or states
- Suitable for **fully observable** environments
- Very fast but **limited intelligence**

How it works:

If condition → then action

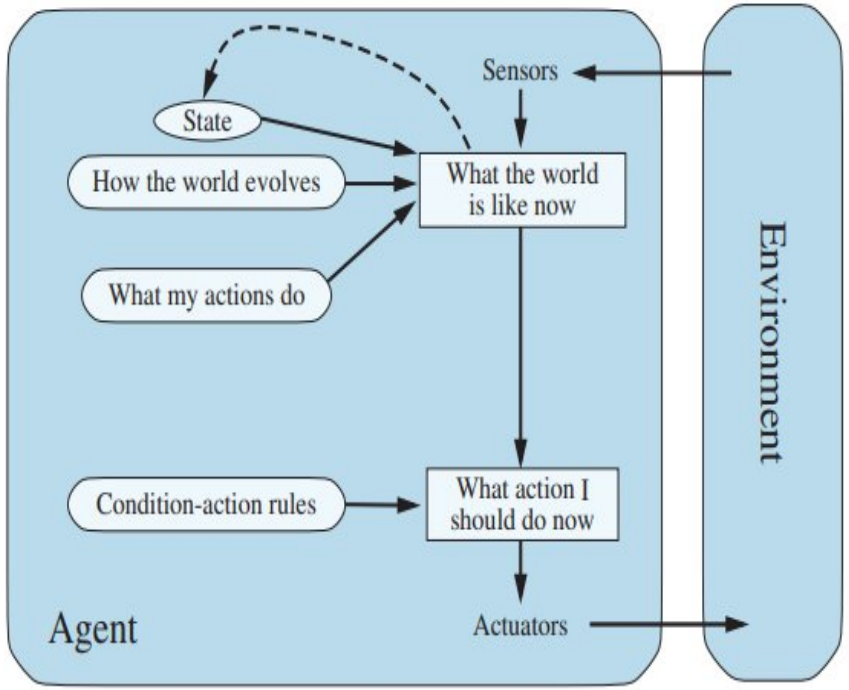
Reflex agents with State (Model-based reflex agents)

About Reflex Agent: These agents maintain some internal state to track aspects of the world that are not immediately perceptible.

Working Principle: They use a model of the world to update their internal state and decide actions.

Advantage: Can handle partially observable environments.

Example: A robot that remembers the layout of a room and updates its knowledge after each movement.



Reflex agents with State (Model-based reflex agents)

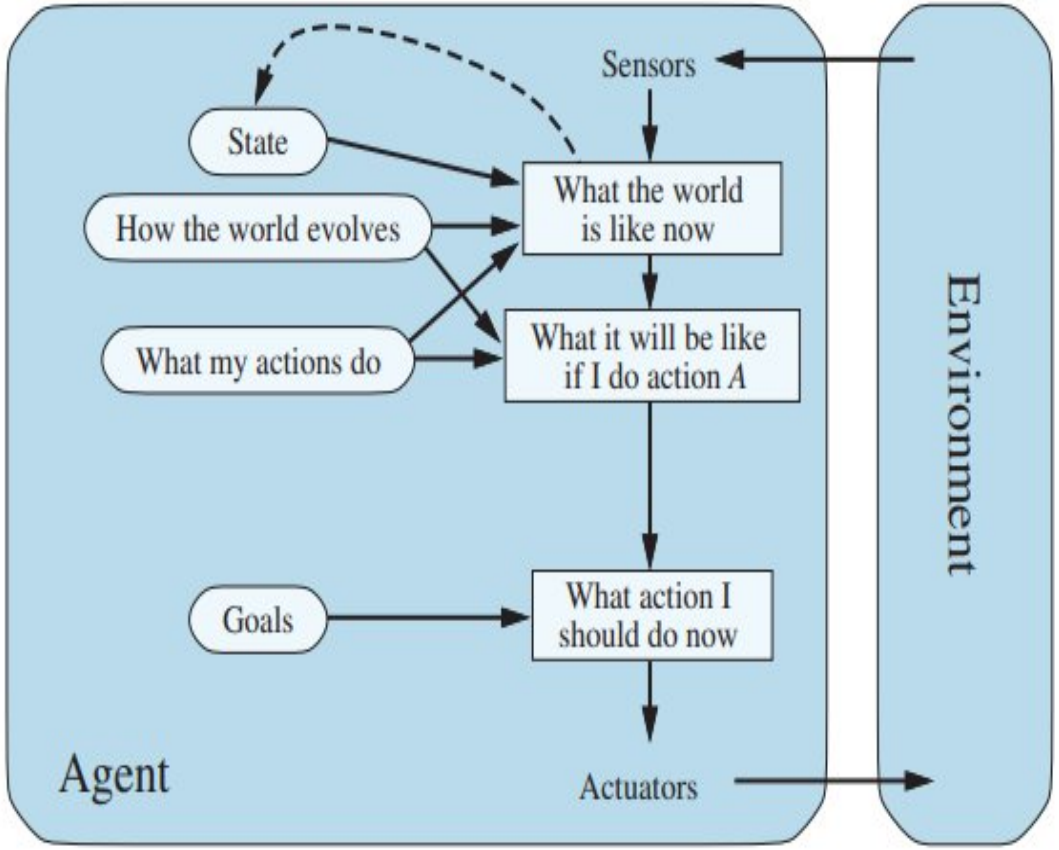
Key Points

- Maintains an **internal state (memory)**
- Uses a **model of the world**
- Can handle **partially observable** environments
- Decisions depend on **current + past percepts**

How it works

Percept → Update internal state → Rule → Action

Goal-based agents:



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

About Goal-based Agent: These agents take actions that help achieve specified goals.

Working Principle: They consider future consequences of actions to choose the one that leads toward the goal.

Advantage: More flexible and capable of planning.

Example: A path-planning robot that avoids obstacles to reach a destination.

Goal-based Agents:

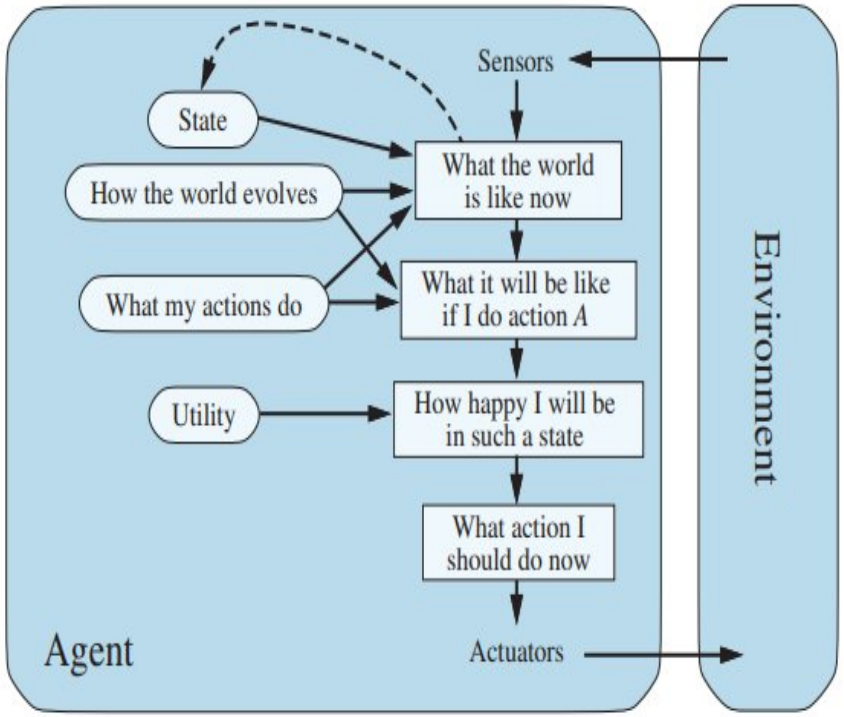
Key Points:

- Actions are chosen to **achieve a specific goal**
- Requires **planning and search**
- Can compare different paths to reach the goal
- More flexible than reflex agents

How it works:

Current state → Possible actions → Choose action that reaches goal

Utility-based agents:



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.

About Utility-based Agent: These agents not only aim to achieve goals but also evaluate the best way to achieve them based on a utility function.

Working Principle: They maximize expected utility rather than just reaching a goal.

Advantage: Can handle trade-offs and uncertainty better.

Example: A self-driving car choosing the safest and most efficient route based on time, distance, and road conditions.

Utility-based Agents:

Key Points:

- Chooses actions that **maximize utility (happiness, profit, comfort, efficiency)**
- Handles **trade-offs** between multiple goals
- Works well under **uncertainty**
- Most **intelligent and flexible**

How it works

For each action → calculate utility → choose highest utility

Good Behavior: The Concept of Rationality

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

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

1. Performance measures

2. Rationality

3. Omniscience, learning, and autonomy.

Good Behavior: The Concept of Rationality

1. Performance measures

- A **rational agent** is defined as one that **performs the right action**, where **"right"** is determined by the **consequences** of the agent's actions, not intentions or rules.
- This idea follows the principle of **consequentialism**, commonly used in AI.
- The performance measure is used to evaluate how desirable the resulting sequence of environment states is after the agent acts. If the outcomes are good according to this measure, the agent is considered rational.

Good Behavior: The Concept of Rationality

1. Performance measures

- A **rational agent** is defined as one that **performs the right action**, where **"right"** is determined by the **consequences** of the agent's actions, not intentions or rules.
- This idea follows the principle of **consequentialism**, commonly used in AI.
- The performance measure is used to evaluate how desirable the resulting sequence of environment states is after the agent acts. If the outcomes are good according to this measure, the agent is considered rational.

we evaluate an agent's behavior by its consequences. When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives. This sequence of actions causes the environment to go through a sequence of states. If the sequence is desirable, then the agent has performed well. This notion of desirability is captured by a performance measure that evaluates any given sequence of environment states.

Good Behavior: The Concept of Rationality

✓ **1. Performance measures**

2. Rationality

An agent's **rationality** depends on four key factors:

1. Performance measure – defines what counts as success.

2. Prior knowledge – what the agent knows about the environment beforehand.

3. Available actions – what the agent is capable of doing.

4. Percept sequence – all information the agent has perceived so far.

A **rational agent** is defined as one that selects an action expected to maximize its performance measure based on its percept history and knowledge.

3. Omniscience, learning, and autonomy.

Good Behavior: The Concept of Rationality

✓ 1. Performance measures

2. Rationality

An agent's **rationality** depends on four key factors:

1. Performance measure – defines what counts as success.

2. Prior knowledge – what the agent knows about the environment beforehand.

3. Available actions – what the agent is capable of doing.

4. Percept sequence – all information the agent has perceived so far.

A **rational agent** is defined as one that selects an action expected to maximize its performance measure based on its percept history and knowledge.

Using the example of a **vacuum-cleaner agent**, it is rational **only if** its actions align with the given environment conditions and performance measure. In a basic setup (clean if dirty, move otherwise), the agent is rational. However, in altered conditions—like a penalty for movement or dynamic dirt—it may act irrationally unless adapted to those changes. Thus, rationality is **context-dependent**, and agents must be designed accordingly.

3. Omniscience, learning, and autonomy.

Good Behavior: The Concept of Rationality

- ✓ 1. Performance measures
- ✓ 2. Rationality

3. Omniscience, learning, and autonomy

- ❖ **Rationality is not perfection**: Rational agents act based on expected outcomes, not guaranteed results-omniscience is unrealistic.
- ❖ **Information gathering** is part of rationality; agents should take actions (e.g., looking before crossing) to improve future decisions.
- ❖ **Learning** enhances rationality by allowing agents to adapt and improve based on experience.
- ❖ **Autonomy** means relying on one's own percepts and learning, not just pre-programmed knowledge.
- ❖ **Non-learning agents** (like the dung beetle or sphex wasp) fail in changing environments, showing the need for adaptive behavior.

➢ A good design blends **initial guidance with learning ability**, enabling agents to operate effectively in varied situations.

Task Environments:

Task environments in AI can be categorized along several key dimensions, which guide agent design:

1. Fully vs. Partially Observable:

Fully observable: Agent has access to complete environment state.

Partially observable: Some relevant information is hidden.

2. Single vs. Multiagent:

Single-agent: No other active agents (e.g., solitaire).

Multiagent: Multiple agents may compete or cooperate (e.g., chess, taxi driving).

3. Deterministic vs. Nondeterministic (Stochastic):

Deterministic: Next state fully determined by current state and action.

Nondeterministic: Outcomes are uncertain; *stochastic* if probabilities are known.

Task Environments:

Task environments in AI can be categorized along several key dimensions, which guide agent design:

4. Episodic vs. Sequential:

Episodic: Each decision is independent of previous ones.

Sequential: Current actions affect future decisions (e.g., chess, driving).

5. Static vs. Dynamic (or Semidynamic):

Static: Environment does not change during deliberation.

Dynamic: Environment changes over time.

Semidynamic: Environment stays static, but performance score changes with time.

6. Discrete vs. Continuous:

Discrete: Finite percepts/actions/states (e.g., chess).

Continuous: Involves real-valued variables and smooth changes (e.g., taxi driving).

7. Known vs. Unknown:

Known: Agent knows outcome rules.

Unknown: Agent must learn how actions affect the environment

The Structure of Agents:

Agent = Architecture + Program.

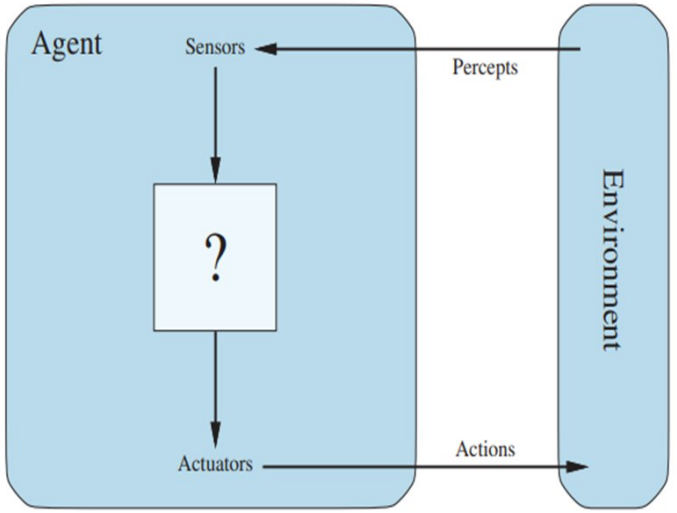
Agent Program: An agent program that implements the agent function — Agent program the mapping from percepts to actions.

Or A software component that maps percepts (inputs from the environment) to actions.

Agent Architecture: The hardware system (sensors + actuators) on which the program runs.

Role of Agent Architecture

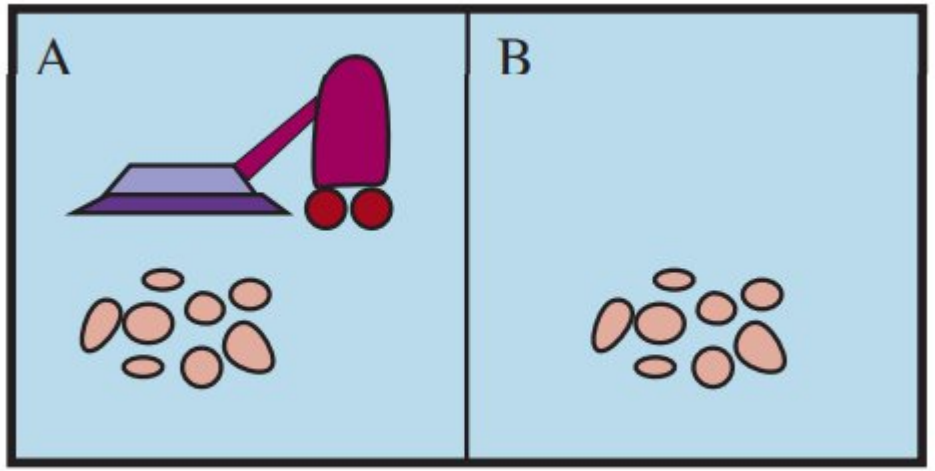
- **Receives percepts** from the environment via **sensors**.
- **Runs the agent program**, which processes the percepts.
- **Executes the selected actions** through **actuators**.



Agent programs:

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
⋮	⋮

Partial tabulation of a simple agent function for the vacuum-cleaner world shown in the right-side figure. 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.



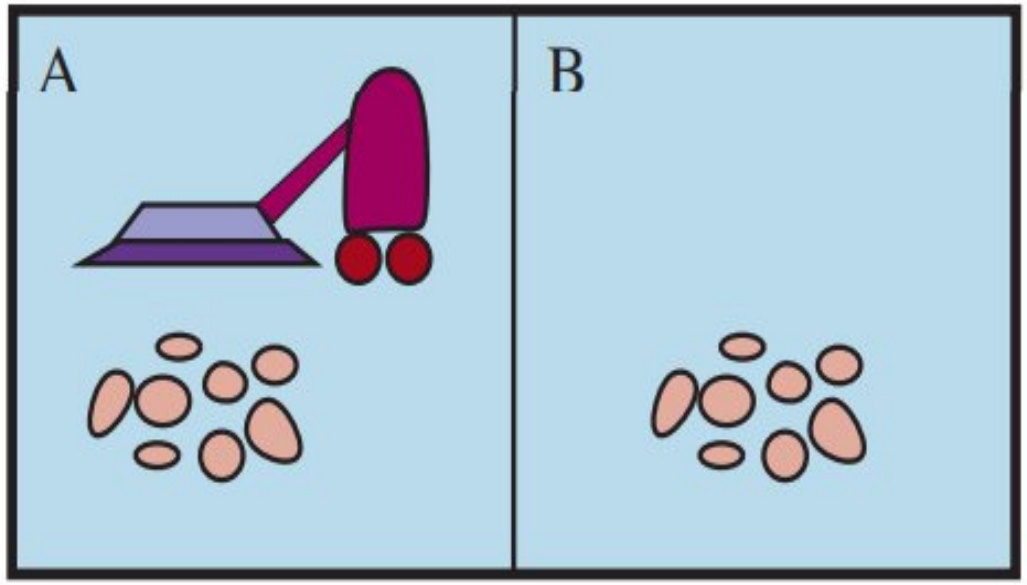
A vacuum-cleaner world with just two locations. Each location can be clean or dirty, and the agent can move left or right and can clean the square that it occupies. Different versions of the vacuum world allow for different rules about what the agent can perceive, whether its actions always succeed,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

Agent programs:

```
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 the previous slide.

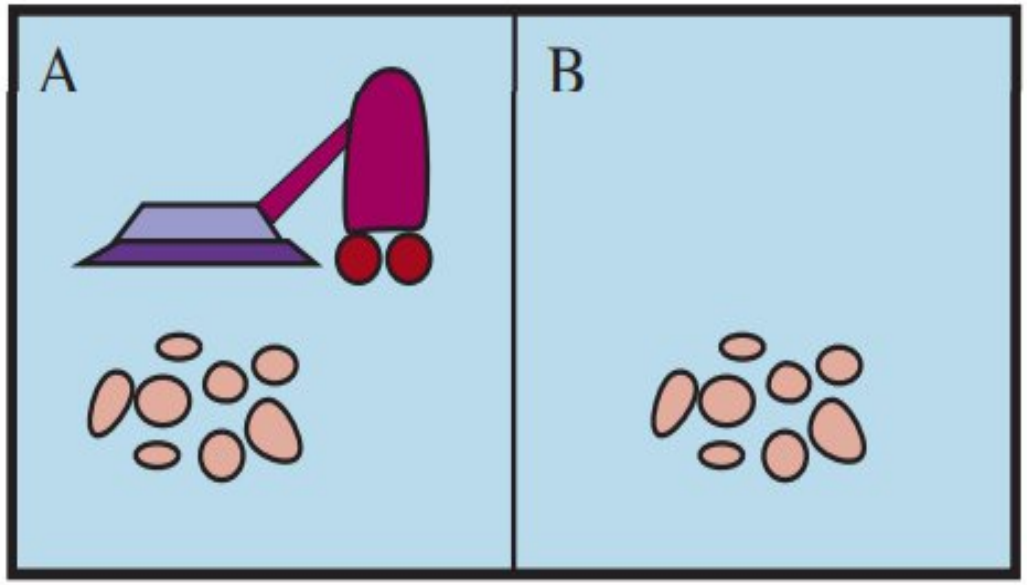


Agent programs:

```
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
```

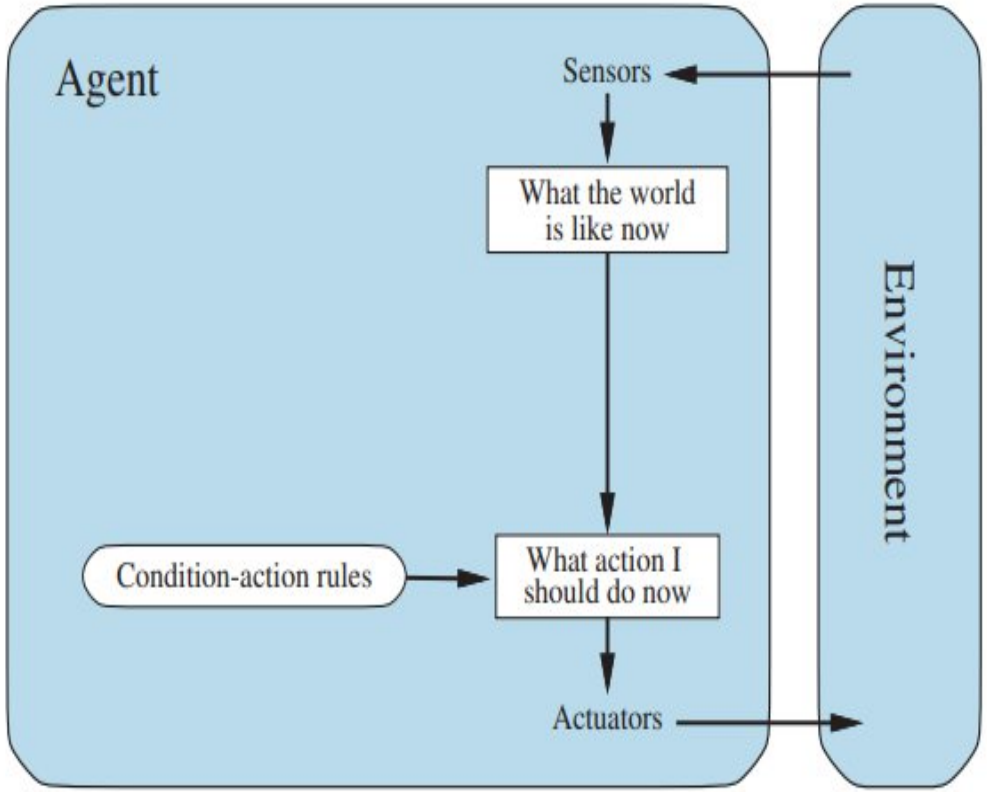
- The agent program for a simple reflex agent in the two-location vacuum environment. This program implements the agent function tabulated in the previous slide.



Agent Programs: - Simple reflex agents

```
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
```



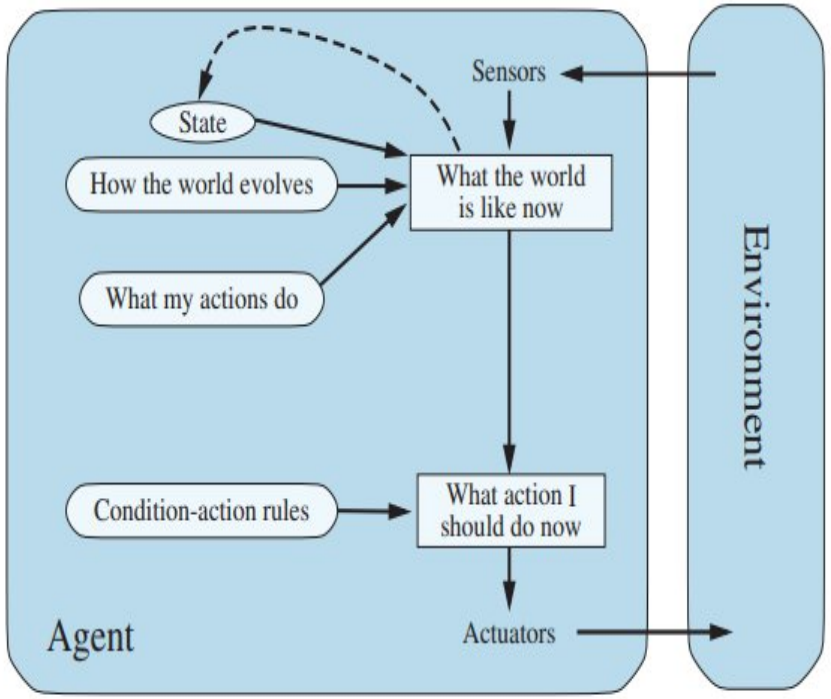
- A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

Agent Programs: - Model-based reflex agents

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.

Learning Agents

"A learning agent is an intelligent agent **that can improve its performance over time by learning from its interactions with the environment**"

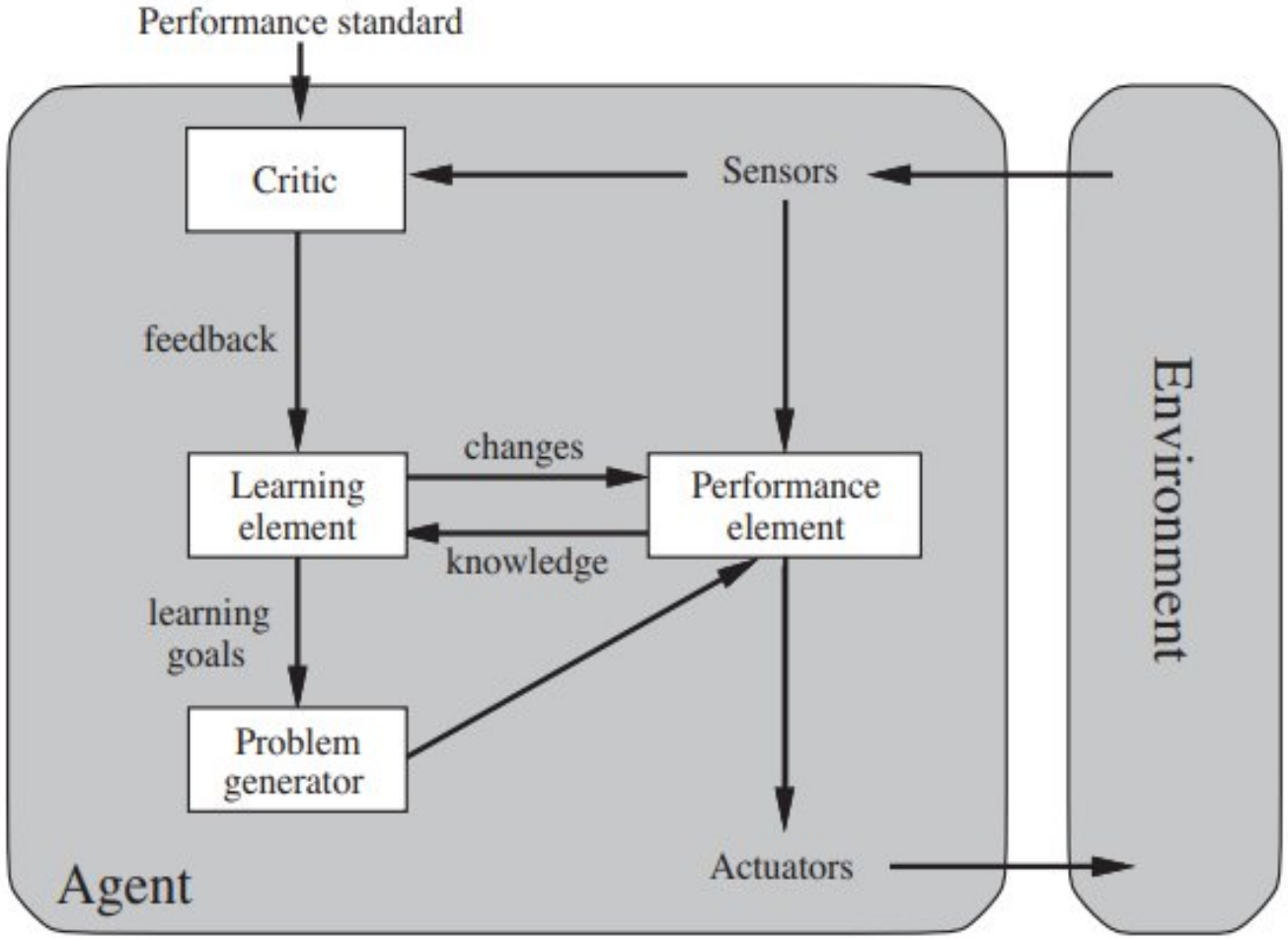
"**A learning agent improves its performance over time by adapting to new experiences and data.**" By IBM

Learning agents:

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."

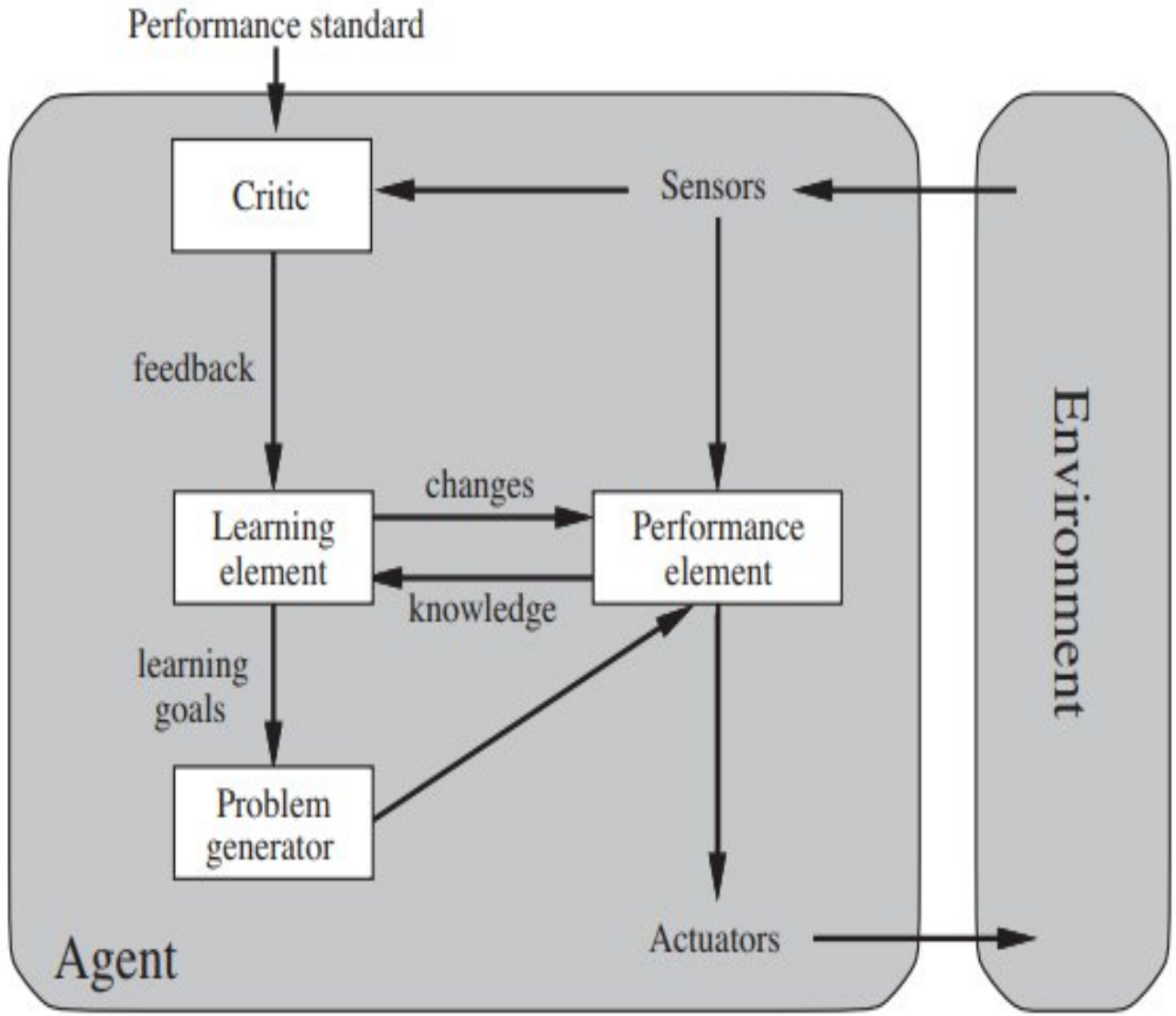
➢ **The method he proposes is to build learning machines and then to teach them. In many areas of AI, this is now the preferred method for creating state-of-the-art systems.**



Learning agents: components

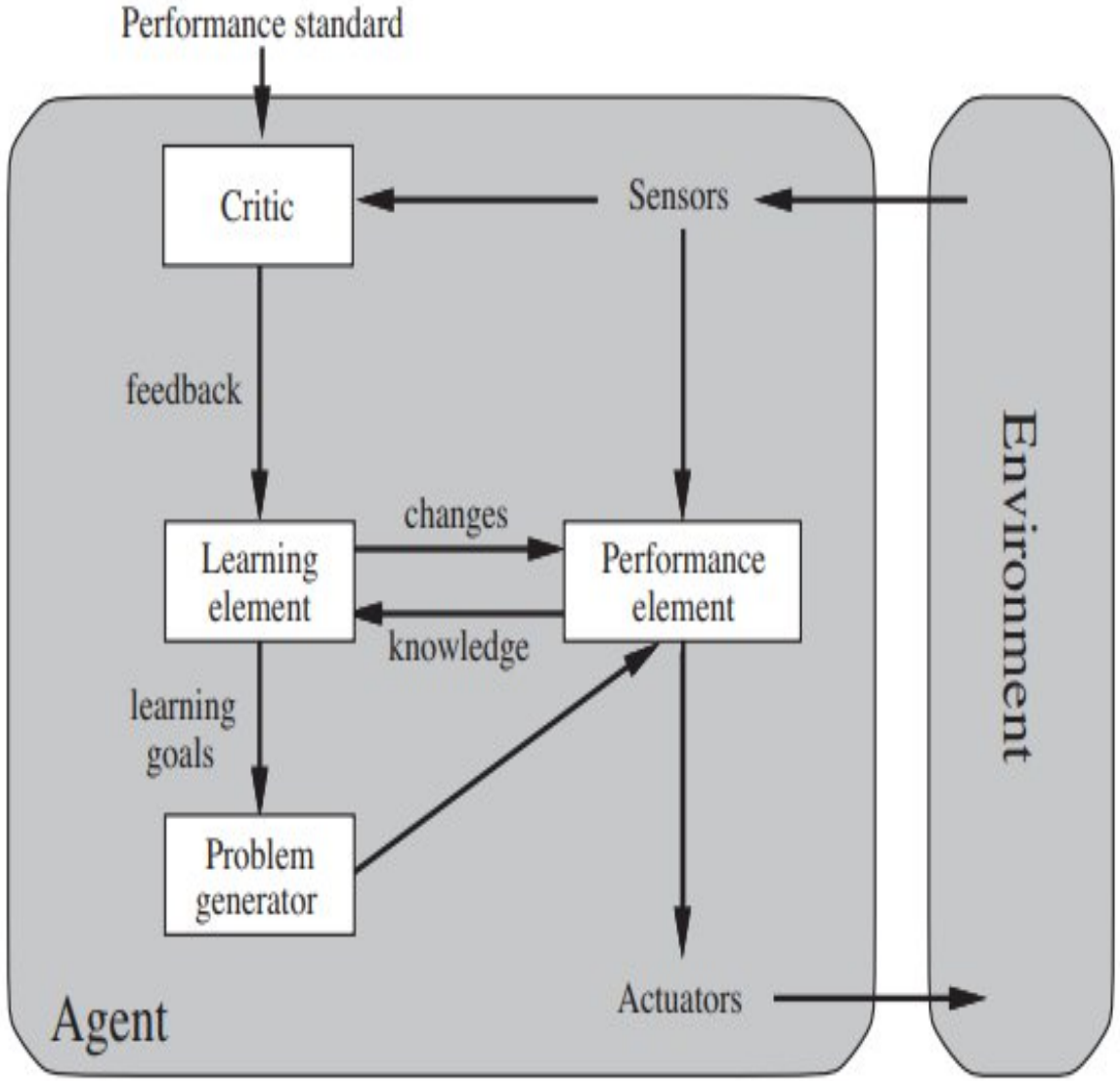
- A learning agent can be divided into four conceptual components, as shown in the figure.

1. **Learning element**
2. **Performance element**
3. **Critic**
4. **Problem Generator**



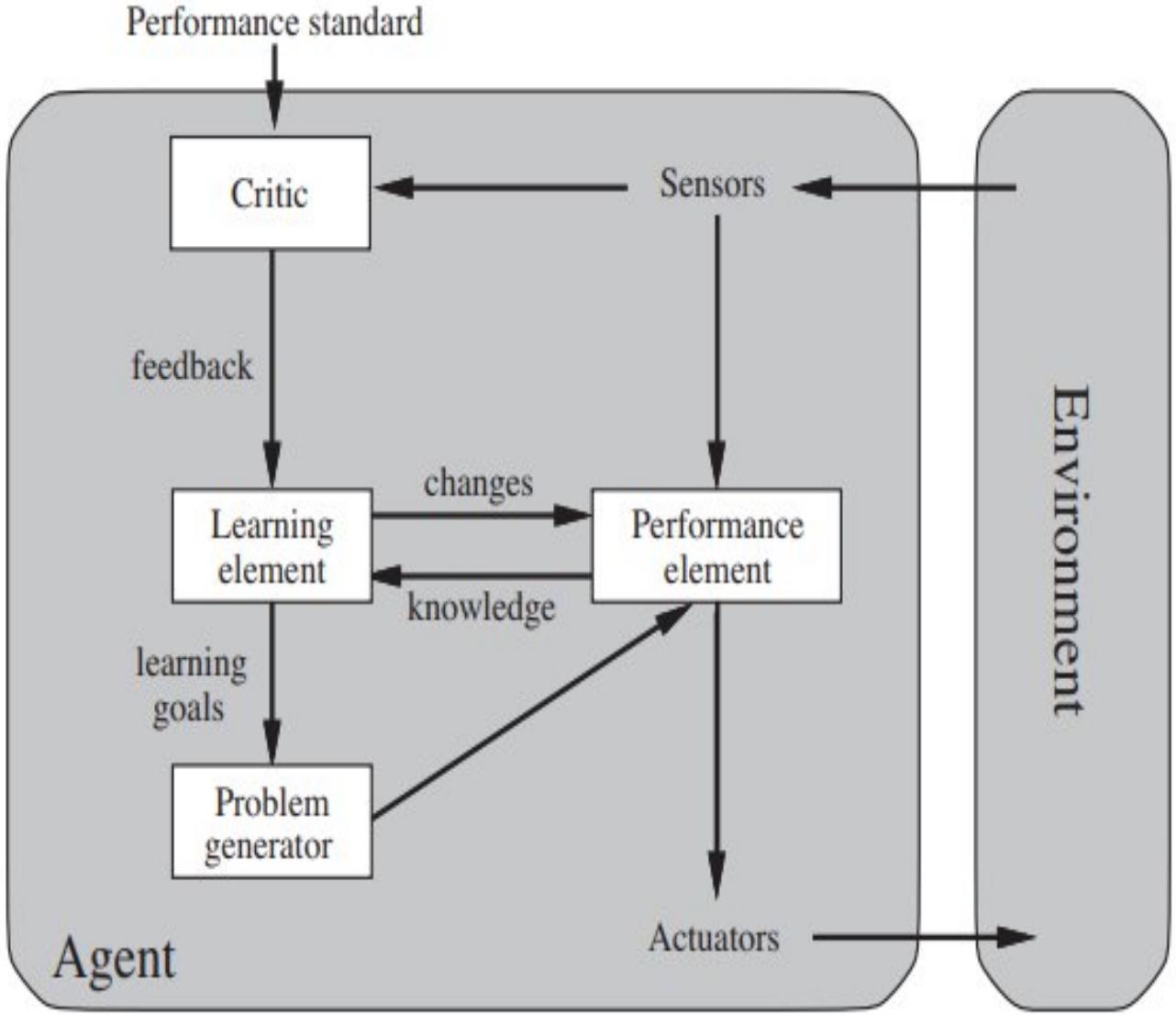
Learning agents: components

- The **learning element**, which is responsible for making improvements.
- The **performance element**, which is responsible for selecting external actions. The performance element is what we have previously considered to be the entire agent: it 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.



Learning agents: components

- The last component of the learning agent is the **problem generator**. It is responsible for suggesting actions that will lead to new and informative experiences.



Learning agents By IBM: https://www.ibm.com/think/topics/ai-agent-learning

A learning agent improves its performance over time **by adapting to new experiences and data.**

Other AI agents work with predefined rules or models, while learning agents continuously update their behavior based on feedback from the environment.

This allows them to enhance their decision-making abilities and perform better in dynamic and uncertain situations. Learning agents represent the full potential of AI tools to handle multistep problem-solving workloads with minimal human intervention.

Learning agents typically consist of **six key components**:

Sensors or perceptors: Perceive the agent's environment and send data to the agent to enable decision-making and behavioral adjustments.

Actuators: The means through which the agent interacts with its environment, such as API integrations.

Performance element: Makes informed decisions based on a knowledge base.

Learning element: Adjusts and improves the agent's knowledge based on feedback and experience.

Critic: Evaluates the agent's actions and provides feedback, often in the form of rewards or penalties.

Problem generator: Suggests exploratory actions to help the agent discover new strategies and improve its learning.

How the components of agent programs work:

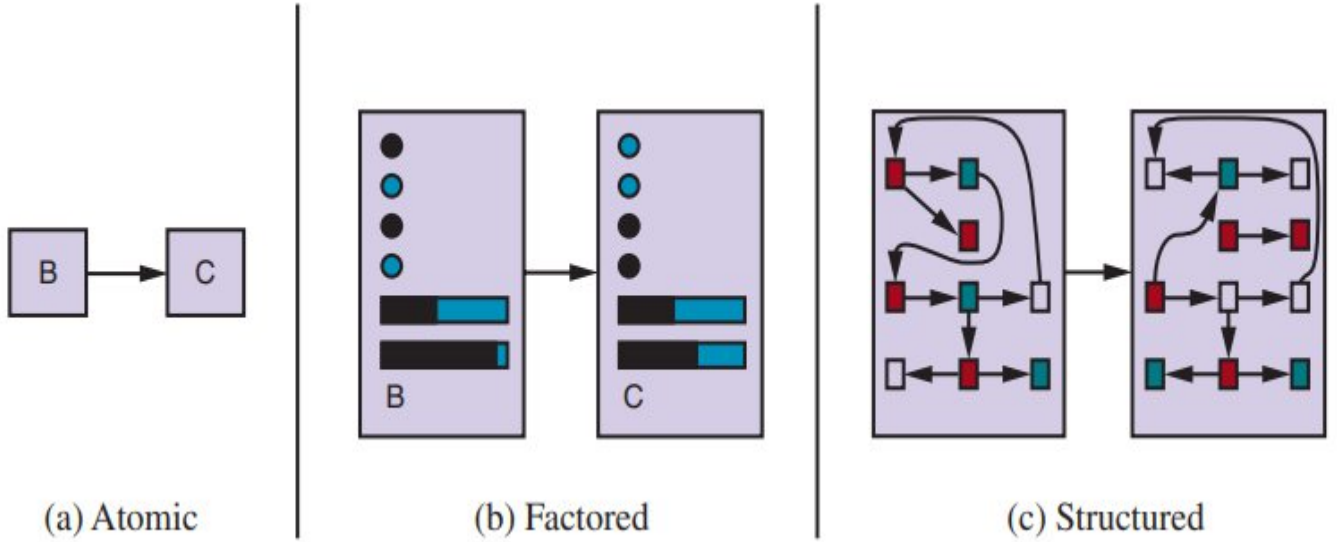
"What is the world like now?"

"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?"



Three ways to represent states and the transitions between them.

(a) Atomic representation: a state (such as B or C) is a black box with no internal structure;

(b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, realvalued, or one of a fixed set of symbols.

(c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

Atomic Representation

Diagram: B → C

Meaning

- The state is treated as a **single indivisible unit**.
- No internal structure is visible to the agent.
-Agent only knows: current state B becomes state C.

Characteristics

- No variables
- No attributes
- No relationships
- Just state-to-state transition

Example

- Chess board treated as a single position ID
- Black-box representation

Used in

- Simple search problems
- Basic reflex agents
- Finite State Machines

$State = s_i$(nointernaldetails)

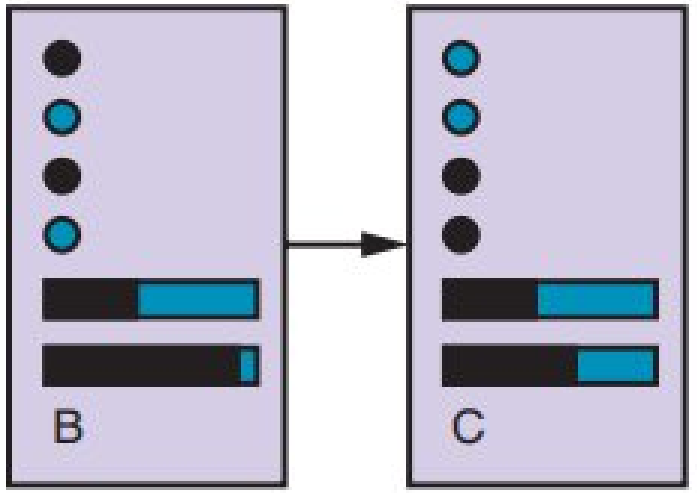
Factored Representation

State = **B**, it is represented as a set of variables:

State **B** = (x_1, x_2, x_3, x_4, x_5)

State = **C**, it is represented as a set of variables:

State **B** = (y_1, y_2, y_3, y_4, y_5)



(b) Factored

When the system moves from **state B to state C**, some variables **change their values**.

Factored Representation

Example: Imagine a robot in a room.

The robot's state may include several variables:

Variable	Meaning
Location	Where the robot is
Battery	Battery level
ObjectHeld	Whether robot holds an object
DoorStatus	Open or closed

State B might be:

Location = Room1

Battery = 70%

ObjectHeld = No

DoorStatus = Closed

After an action, the robot moves to **State C**:

Location = Room2

Battery = 65%

ObjectHeld = Yes

DoorStatus = Closed

Why Factored Representation Is Useful

1.More detailed description of states

2.Allows **logical reasoning**

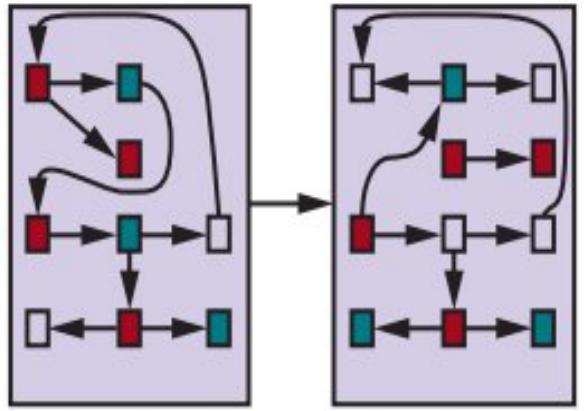
3.Helps in **probabilistic models**

4. Used in many AI systems such as:
   1. Planning
   2. Bayesian Networks
   3. Markov Decision Processes
   4. Reinforcement Learning

Structured Representation

In this representation, a state is described using:

- **Objects**
- **Attributes**
- **Relationships between objects**



(c) Structured

Unlike atomic or factored representation, structured representation **shows how different components are connected to each other**.

Structured Representation

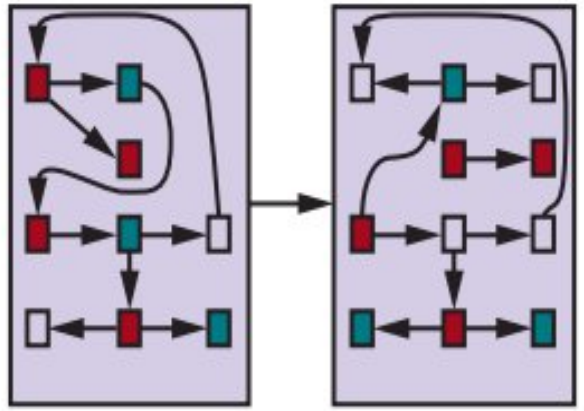
The diagram contains **two states** (left and right) connected by an **action or transition**.

Inside each state:

- **Small colored squares** represent **objects or entities**
- **Arrows** represent **relationships or interactions between objects**
- The **entire network** represents the structure of the environment

When the system moves from the **left state to the right state**, the **relationships between objects change**.

So the state is not just a list of variables but a **structured network of related objects**.



(c) Structured

Structured Representation

Example: Blocks World

A classic AI example is the **Blocks World problem**.

Suppose we have blocks: A, B, C

Relationships may be:	Meaning:
On(A, B)	- Block **A is on B**
On(B, Table)	- Block **B is on the table**
Clear(A)	**- A and C are free on top**
Clear(C)	

If the robot moves **block A**, the relationships change:

On(A, Table)

On(B, Table)

Clear(A)

This change in **relationships** is exactly what the structured diagram represents.

Structured Representation: Why and Where

Why Structured Representation is Powerful

Structured representation allows AI systems to:

- Understand **relationships between objects**
- Perform **logical reasoning**
- Represent **complex environments**

Where It Is Used

Structured representation is used in:

- **Knowledge Representation**
- **Semantic Networks**
- **First Order Logic**
- **Knowledge Graphs**
- **Natural Language Processing**
- **Expert Systems**

For more details, please go to Chapter 2, pp. 54-78 of the following Book.

Book 1: "Artificial Intelligence: A Modern Approach", Fourth Edition, by Stuart Russell and Peter Norvig.

Link: https://people.engr.tamu.edu/guni/csce625/slides/AI.pdf

Link: https://aima.cs.berkeley.edu/

Post your doubts:
https://www.gcjana.in/courses/shardauniversity/2501/CSE472/#Post-doubts