Intelligent Agents

Outline

- Agents and environments
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Types of Environment
- Types of Agent

Agents and Environments

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators
- Human agent:
  - eyes, ears, and other organs for sensors;
  - hands, legs, mouth, and other body parts for actuators
- Robotic agent:
  - cameras and infrared range finders for sensors;
  - various motors for actuators
- Software Agent:
  - Keystrokes, files, network contents;
  - Displaying results, writing files, transmitting data
- Percept vs Percept Sequence:
  - Percept: Agent's perceptual inputs at any given instant
  - Percept Sequence: Complete history of everything that the agent has ever perceived.

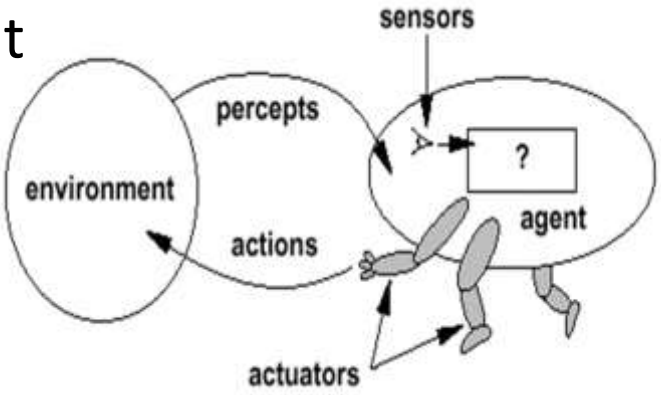
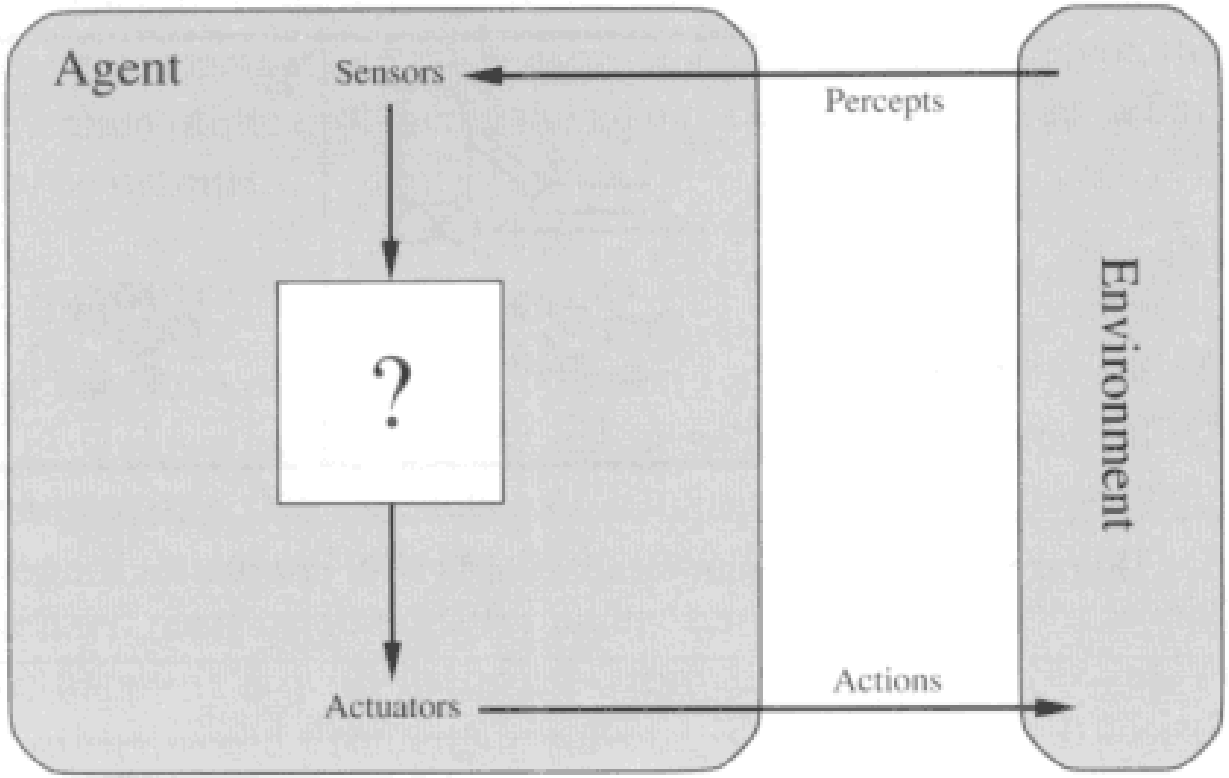


Diagram of an agent

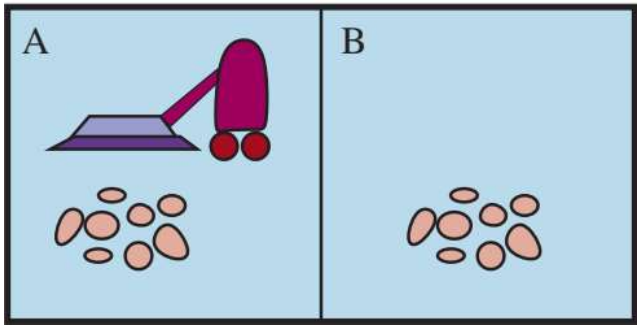
- Diagram showing how an agent interact with the environment using sensors and actuators



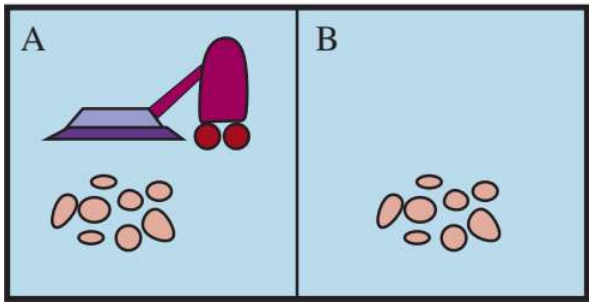
Agent Function vs Agent Program

To answer the question, What does an agent do?

- Agent's behavior is *mathematically* described by
  - **Agent function:** A function mapping any given percept sequence to an action
- *Practically* it is described by:
  - An **agent program:** The real implementation
- Example (Agent Function Vs Agent Program)
- The Vacuum Cleaner World
  - Has just two locations: squares A and B
- The Vacuum agent
  - Perceives which square it is in and whether there is dirt in the square
  - Can choose to move left, move right, suck up the dirty or do nothing
- Agent Function
  - If the current square is dirty, then suck; otherwise move to the other square



Agent Function vs Agent Program



A vacuum-cleaner world with just two locations.

Agent Function

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

Agent Program

Function Reflex-Vacuum-Agent([*location,status*]) **return** an action

If *status = Dirty* **then return** *Suck*

else if *location = A* **then return** *Right*

else if *location = B* **then return** *left*

Agent Function vs Agent Program

- Agent function: describes agent's behavior by mapping any given percept sequence to an action
  - To describe any given agent, we have to tabulate the agent function- and this will typically be a very large table (potentially infinite)
  - The table, in principle can be constructed by **trying out all possible percept sequences** and recording which actions the agent does in response
  - The table is **external characterization** of the agent
  - Agent function is abstract mathematical description
- Agent program: is an internal implementation of the agent function for an artificial agent
  - It is a **concrete implementation** running within some physical system

Good Behavior: The Concept of Rationality

It is learnt that we should design agents that “act rational”

- How do we define ‘acting rationally’ so we can write programs?
- Should we consider the environment where the agent will be deployed?

- A rational agent is one that does the right thing
  - In other words, every entry in the table for the agent function is filled out correctly
- What does it mean to do the right thing?
  - We consider the consequences of the agent’s behavior
  - When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives
  - The sequence of actions causes the environment to go through a sequence of states
  - If this sequence of environment is desirable, the agent has performed well

Agent

Environment

Good Behavior: The Concept of Rationality

- We are interested in environment states not agent states
  - We should **not** define success in terms of the **agent's opinion**
  - The agent could achieve a perfect rationality simply by deluding itself that its performance was perfect
  - Human agents, for example, are notorious for "sour grapes" – believing that they did not really want something after not getting it.

Rationality

- What is rational 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.
- 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.

Rationality

- Let us assume the following for the Vacuum Cleaner example
  - Performance measure: awards one point for each clean square at each time step, over a “lifetime” of 1000 time steps.
  - Prior Knowledge: the agent knows that there are two squares but it does not know the dirt distribution
  - Agent Actions: available actions are left, right and suck dirt
  - Percept Sequence: agent correctly perceives its location and whether that location contains dirt
- What the vacuum-cleaner agent does:
  - Cleans a square if it is dirty and moves to the other square if not
  - Is the vacuum cleaner agent rational? Is performance measure robust?
  - What will the agent do after all dirt is cleaned up? Oscillate back and forth

Rational agents

- Rationality is distinct from omniscience (all-knowing with infinite knowledge, state of knowing everything)
- Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering, exploration)
- An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt)

Specifying the Task Environment

- In designing an agent, the first step must always be to specify the task environment as fully as possible
- The task environment is the description of Performance, Environment, Actuators and Sensors (PEAS)
- Example: A fully automated taxi currently is somewhat beyond the capabilities of existing technology because the full driving task is extremely open-ended

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

Example of PEAS

- **Agent: Medical diagnosis system**
  - Performance measure: Healthy patient, minimize costs
  - Environment: Patient, hospital, staff
  - Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
  - Sensors: Keyboard (entry of symptoms, findings, patient's answers)
- **Agent: Part-picking robot**
  - Performance measure: Percentage of parts in correct bins
  - Environment: Conveyor belt with parts, bins
  - Actuators: Jointed arm and hand
  - Sensors: Camera, joint angle sensors
- **Agent: Interactive English tutor**
  - Performance measure: Maximize student's score on test
  - Environment: Set of students
  - Actuators: Screen display (exercises, suggestions, corrections)
  - Sensors: Keyboard

Summary

- An **agent** is something that observes and acts in an environment.
- The **agent function** for an agent specifies the action taken by the agent in response to any percept sequence.
- The **performance measure** evaluates the behavior of the agent in an environment.
- A **rational agent** acts so as to maximize the expected value of the performance measure, given the percept sequence it has seen so far.
- A **task environment** specification includes the performance measure, the external environment, the actuators, and the sensors (PEAS).
- In designing an agent, the first step must always be to specify the task environment as fully as possible.

Types of Environment

- An environment in artificial intelligence is the surrounding of the agent. The agent takes input from the environment through sensors and delivers the output to the environment through actuators. There are several types of environments:

  - Fully Observable vs Partially Observable
  - Deterministic vs Stochastic
  - Competitive vs Collaborative
  - Single-agent vs Multi-agent
  - Static vs Dynamic
  - Discrete vs Continuous
  - Known vs Unknown

Fully Observable / Partially Observable

Fully Observable

- An agent can always see the entire state of an environment.
- When an agent sensor is capable to sense or access the complete state of an agent at each point in time, it is said to be a fully observable environment else it is partially observable.
- **Example:**
  - **Chess –** the board is fully observable, and so are the opponent's moves.

Partially Observable

An agent can never see the entire state of an environment

Example:

Card Game – the environment is partially observable because what's in other player hands is not known.

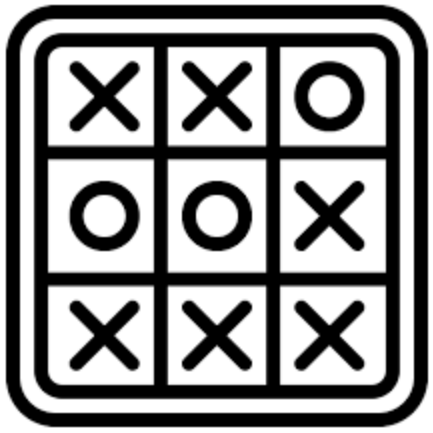
Deterministic / Stochastic

Deterministic

- An agent's current state and selected action can completely determine the next state of the environment
- Example: tic tac toe

Stochastic

- The stochastic environment is random in nature which is not unique and cannot be completely determined by the agent.
- Example: Ludo (Any game that involve dice)



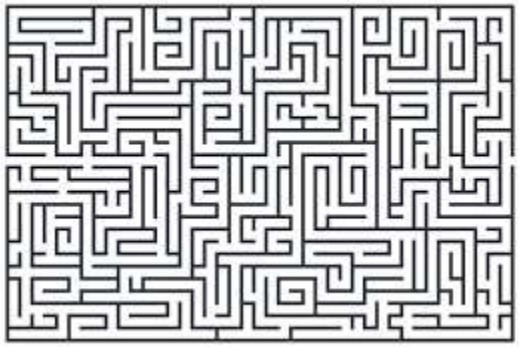
Single-agent / Multi-agent

Single-agent

- If only one agent is involved in an environment, and operating by itself such an environment is called single agent environment
- Example: Maze

Multi-agent

- If multiple agents are operating in an environment, then such an environment is called a multi-agent environment
- Example: Football



Static / Dynamic

Static

- The environment does not change while an agent is acting
- Example: Crossword puzzles

Dynamic

- The environment may change over time
- Example: Taxi Driving (self driving)

Discrete/ Continuous

Discrete

- The environment consists of a finite number of actions that can be deliberated in the environment to obtain the output
- Example: Chess

Continuous

- The environment in which the actions performed cannot be numbered i.e. is not discrete is said to be continuous.
- Example: Taxi Driving (self driving)

Known/ Unknown

Known

- In a known environment, the results for all actions are know to the agent
- Example: card game

Unknown

- In unknown environment, agent needs to learn how it works in order to perform an action.
- Example: A new video game

Structure of Intelligent Agent

- Understanding structure of an Agent requires familiarization with **Architecture** and **Agent programs**.
- **Architecture** is the machinery that the agent executes on. It is a device with sensors and actuators, for example, a robotic car, a camera, a PC.
- **Agent program** is an implementation of an agent function. An agent function is a map from the percept sequence(history of all that an agent has perceived to date) to an action.
- In summary:

Agent = Architecture + Agent Program

Example Agent program (table driven)

function TABLE-DRIVEN-AGENT(*percept*) **returns** an action
  persistent: *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*

Figure 2.7 The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

Types of Agent Programs

- Agents can be grouped into five classes based on their degree of perceived intelligence and capability :
  - Simple Reflex Agents
  - Model-Based Reflex Agents
  - Goal-Based Agents
  - Utility-Based Agents
  - Learning Agent

Simple Reflex Agents

- Simplest kind of agent
- Select actions on the basis of the ***current*** percept
- Ignores the rest of the percept history
- Makes use of **condition–action rule.**

- Condition–Action rule
  - In Condition-Action rule a change to the condition “is” the triggering event.
  - Syntax: **if** *Condition* **then** *Action*
  - Example: **if** *Room-is-dirty* **then** *Clean*.

Simple Reflex Agent

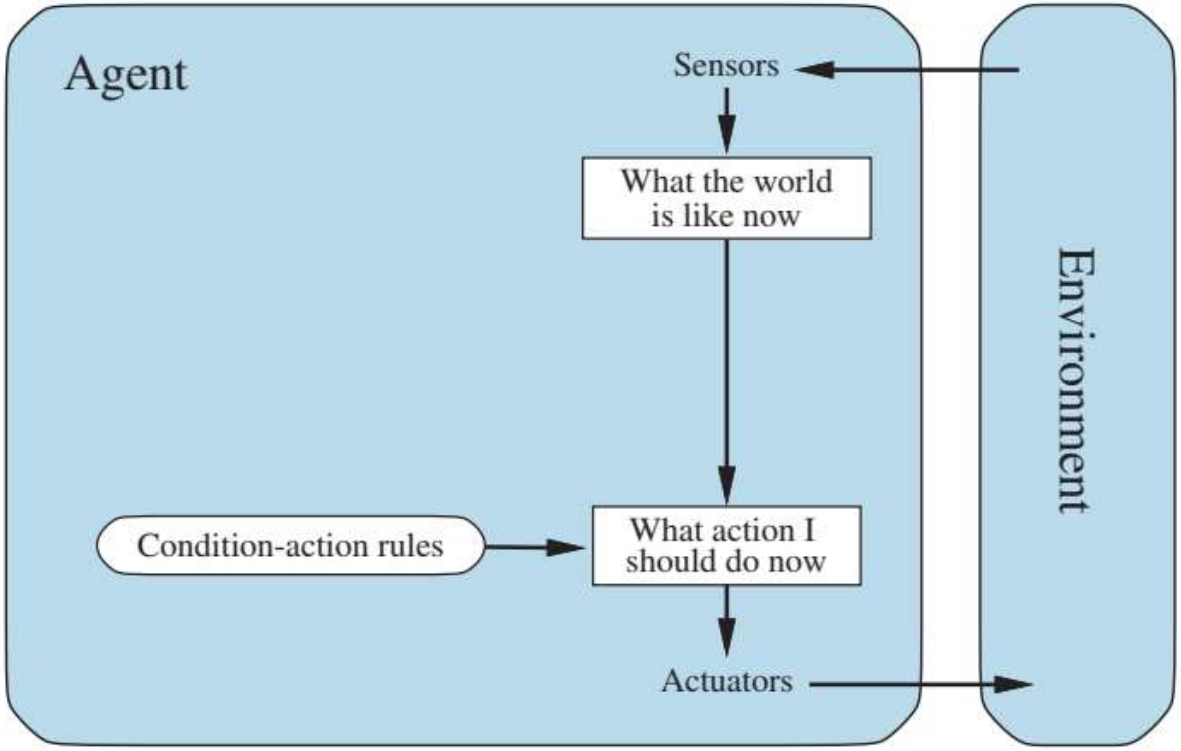


Figure 2.9 Schematic diagram of a simple reflex agent. We use rectangles to denote the current internal state of the agent's decision process, and ovals to represent the background information used in the process.

Simple Reflex Agent

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*

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

Limitations of Simple Reflex Agents

- Have limited intelligence
- Even a little bit of un-observability can cause serious trouble (only works in fully observable environment)
- If there occurs any change in the environment, then the collection of rules need to be updated.
- Infinite loops are often unavoidable for simple reflex agents operating in partially observable environments (Example of a vacuum cleaner with no location sensor)
- Escape is randomization

Model based Reflex agent

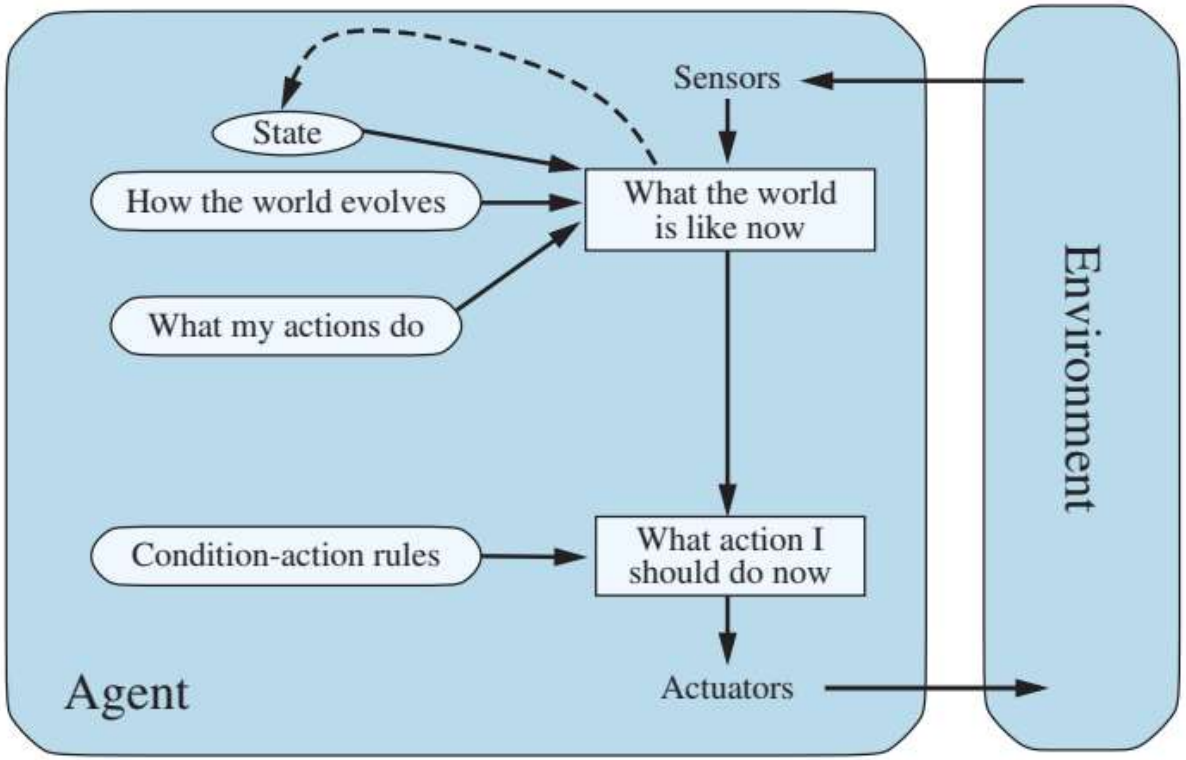


Figure 2.11 A model-based reflex agent.

Model Based Reflex Agents

- Maintains **internal state**
  - the percept history
- Updating internal state information requires:
  - some information about how the world evolves independently of the agent
  - some information about how the agent's own actions affect the world
- This knowledge about "how the world works" is called a **model** of the world
  - An agent that uses such a model is called a **model-based agent**.

Cont.

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*

Figure 2.12 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.

Goal based agents

- Expansion of Model Based Reflex Agent
- Desirable situation -> **GOAL**
- Searching & Planning

Goal based agent

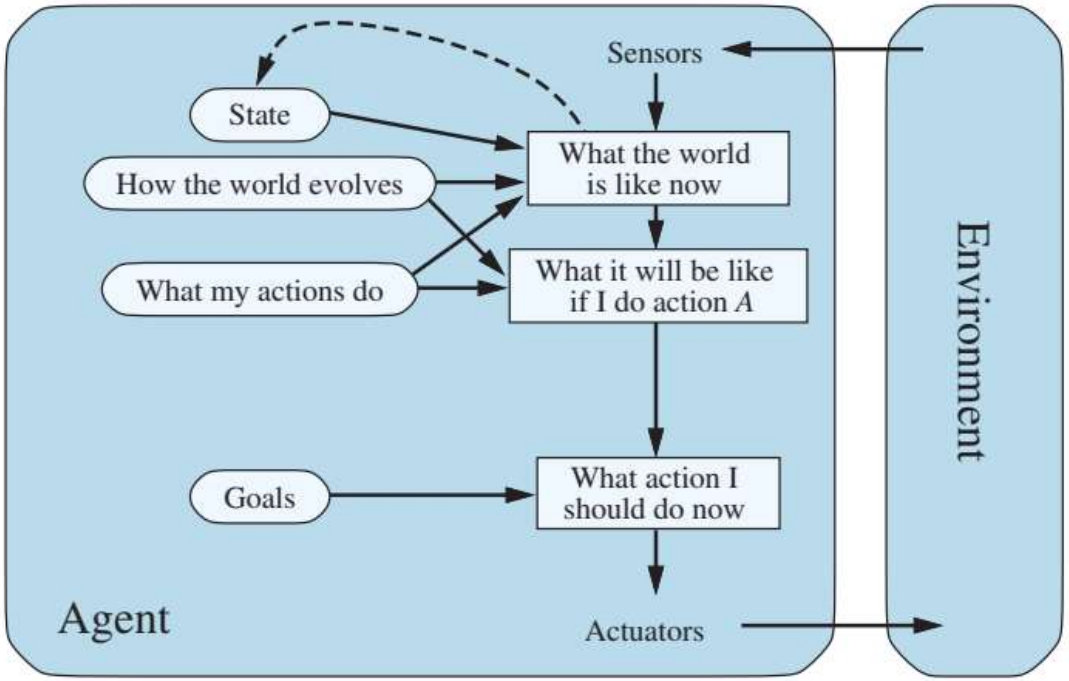
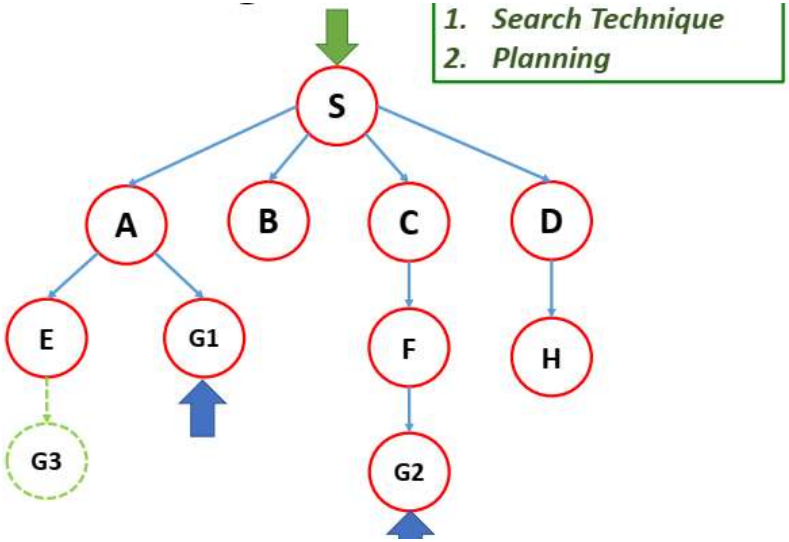


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Cont.

- Examples: Search and planning
- These agents deals with the future
- What will happen if I do such and such
- What will make me happy
- More flexible because the knowledge that supports its decisions is explicit and changeable
- this will automatically alter the agent behavior to suit new environmental conditions
- Example: GPS system



Utility based agent

- Goals alone are not enough to generate high quality behavior in most environments
- Goals just provide a crude binary distinction between “happy” and “unhappy” states.
- Many action sequences will get the taxi to its destination
  - But some are quicker, safer, more reliable, or cheaper than others
- Performance measure assigns a score to any given sequence of environment states
- Utility agents provide the solution which **maximizes** the **performance measure** OR
- 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.

Cont.

Some solutions to goal states are better than others.
Which one is best is given by a utility function.
Which combination of goals is preferred?

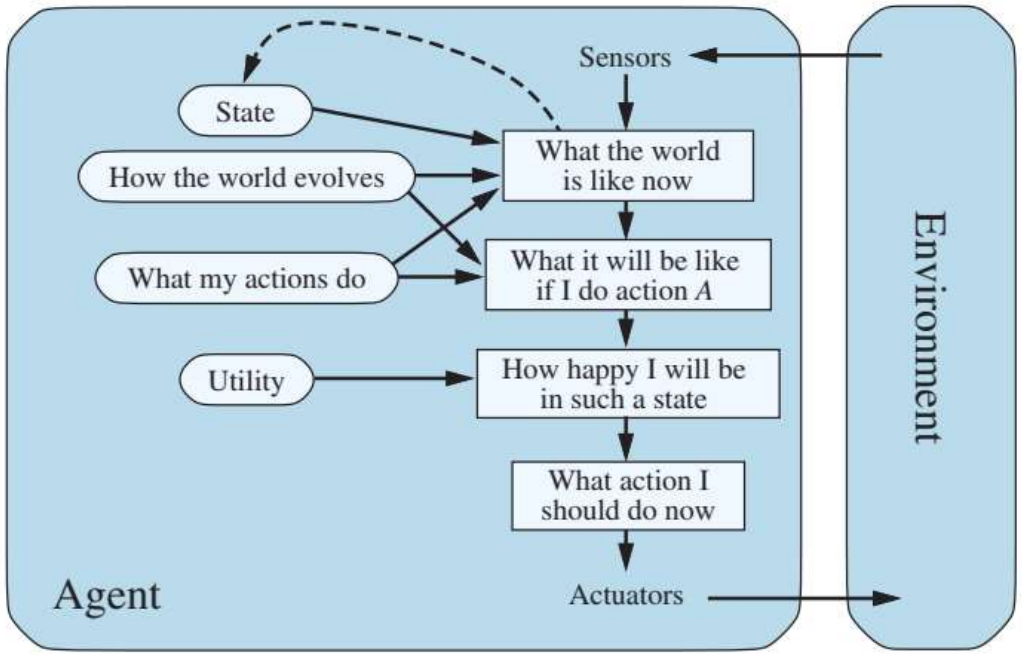


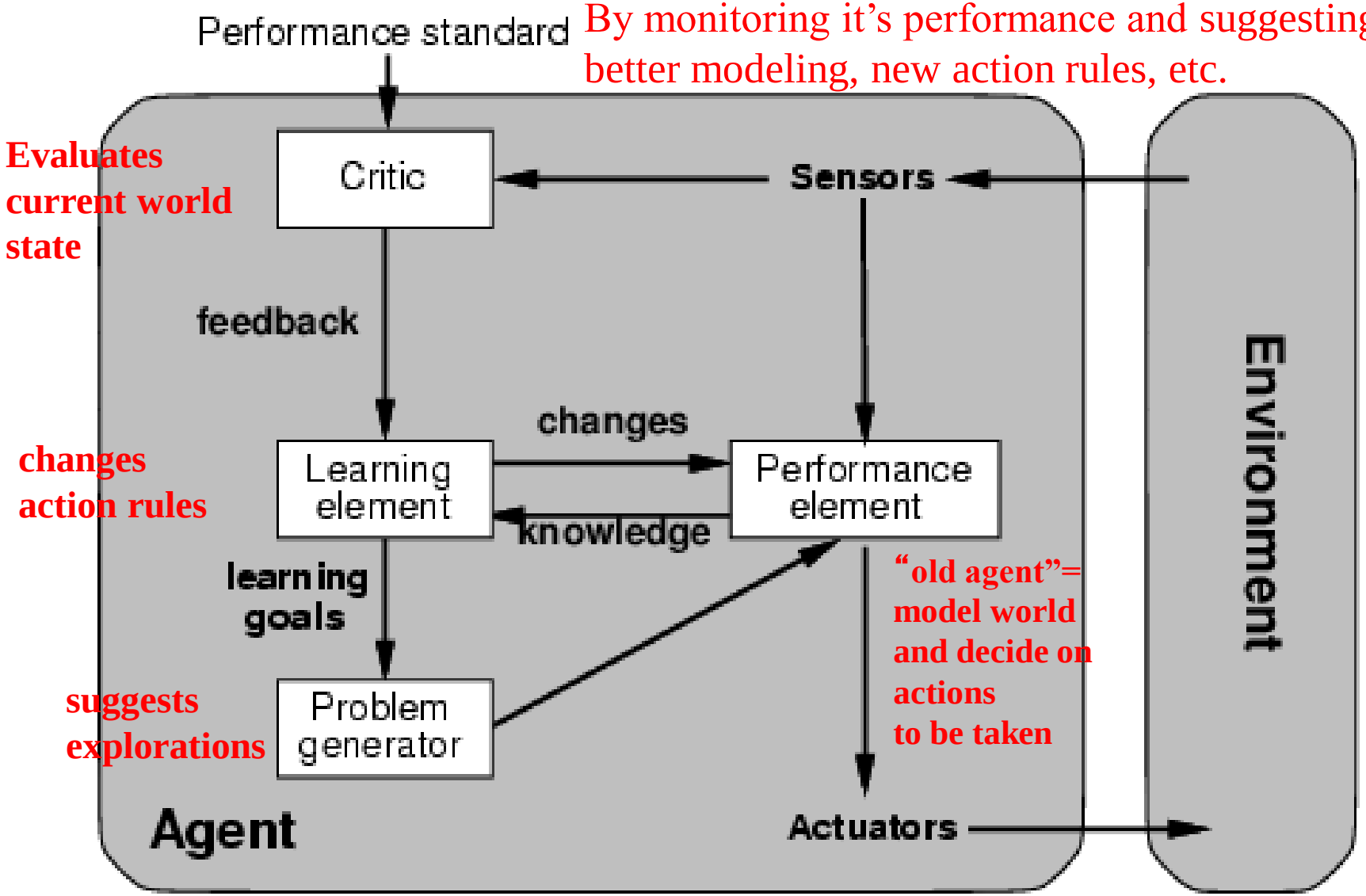
Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

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 agents

How does an agent improve over time?
By monitoring it's performance and suggesting better modeling, new action rules, etc.



How the components of agent programs work

- Agent Program contains components which answers:
  - “What is the world like now?”
  - “What action should I do now?”
  - “What do my actions do?”
- The next question is, “How these components work?”
- We place the representations along an axis of increasing complexity and expressive power; namely:
  - **Atomic**
  - **factored**, and
  - **structured**.

- **Atomic**: where state of the world is indivisible, e.g binary or In (Peshawar)
- **Factored representation** splits up each state into a fixed set of **variables** or **attributes**, each of which can have a **value**. E.g (In Islamabad) = amount of gas, GPS coordinate, route condition
- **structured representation**, in which objects and their various and varying relationships can be described explicitly e.g. TruckAheadBackingIntoDairyFarmDrivewayBlockedByLooseCow with value true or false

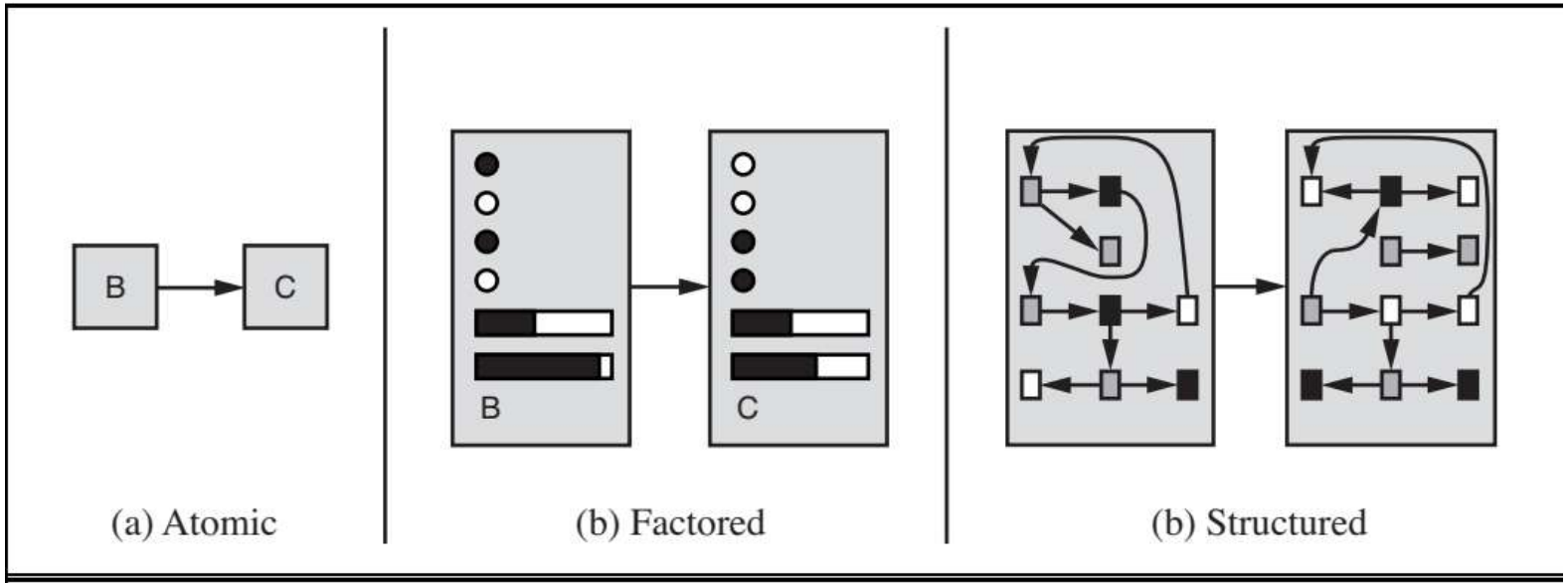


Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed 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.