Artificial Intelligence

By
S. Komal Kaur,
Assistant Professor,
Vasavi College of Engineering,
Hyderabad

Chapter 2

- Intelligent Agents
- Agents and Environments
- Good Behavior: The concept of Rationality
- The Nature of Environments
- The Structure of Agents

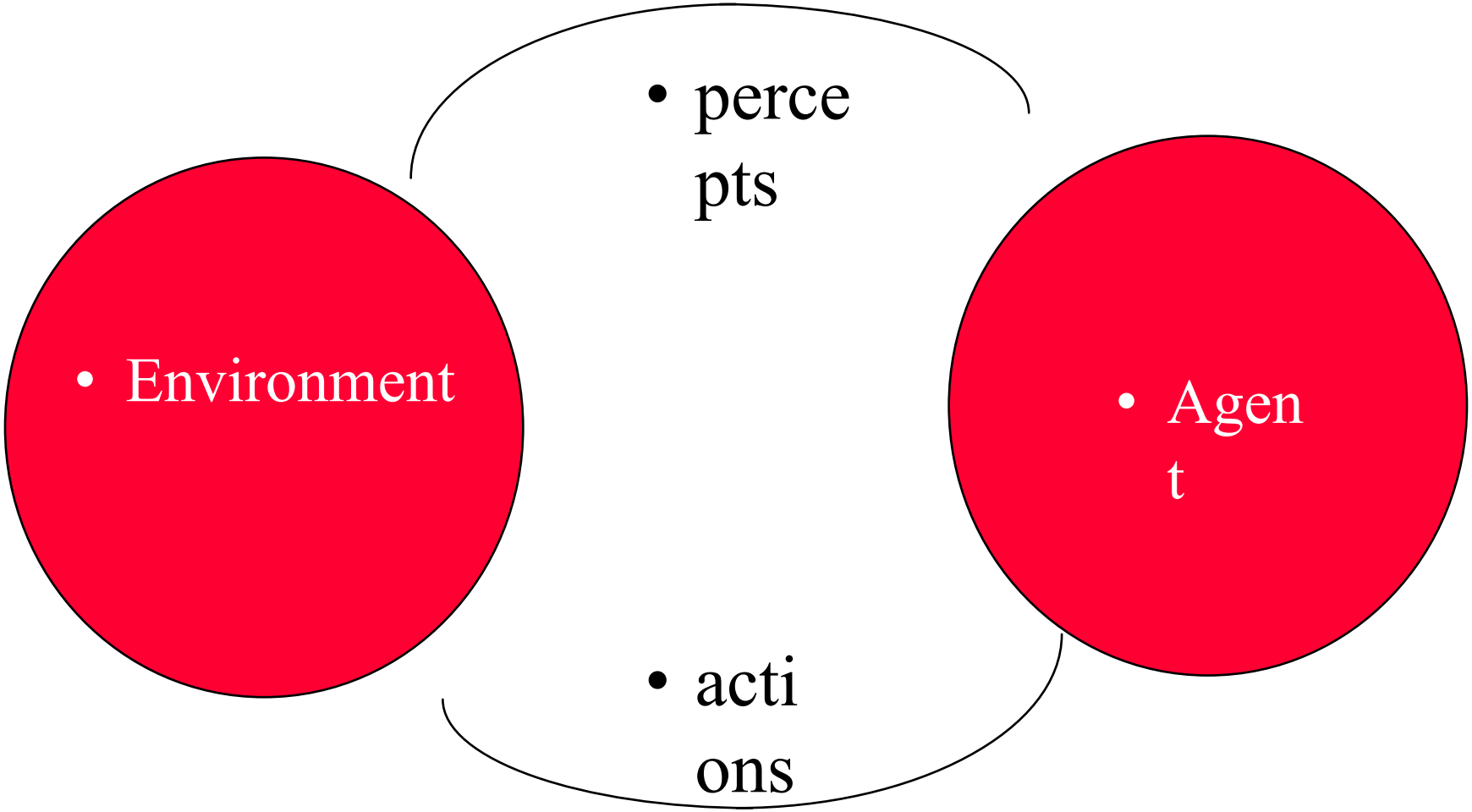
Intelligent Agents

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through 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.
- *An agent's choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn't perceived.*
- An agent's behavior is described by the **agent function** that maps any given percept sequence to an action.

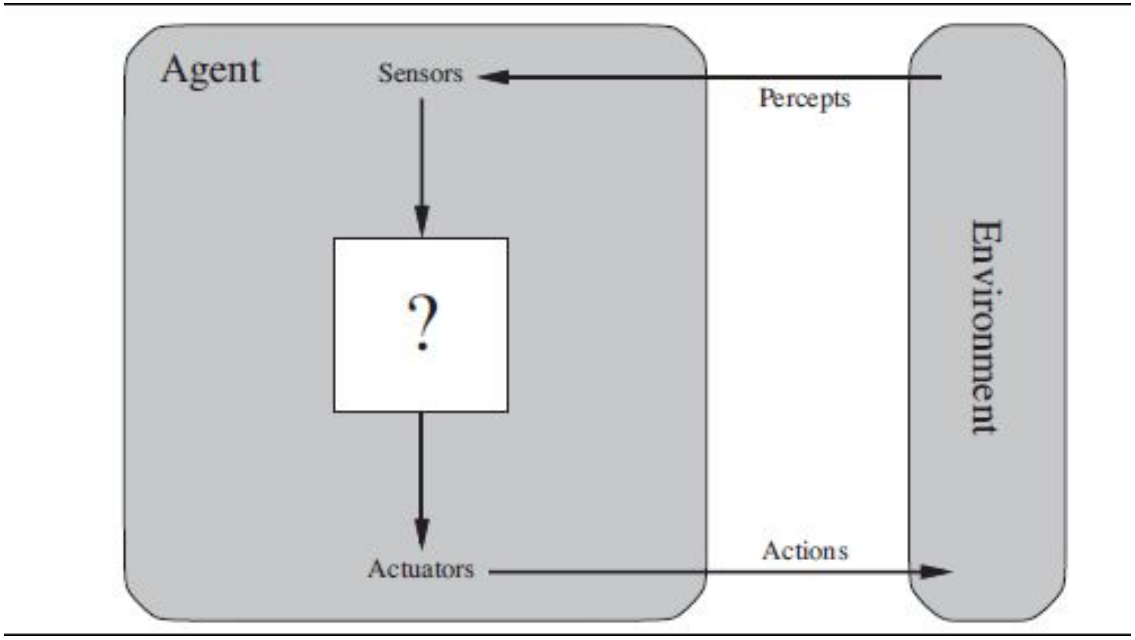
Agent

- We use the term percept to refer to the agent's perceptual inputs at any given instant.
- An agent's percept sequence is the complete history of everything the agent has ever perceived.
- In general, an agent's choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn't perceived

Intelligent Agents



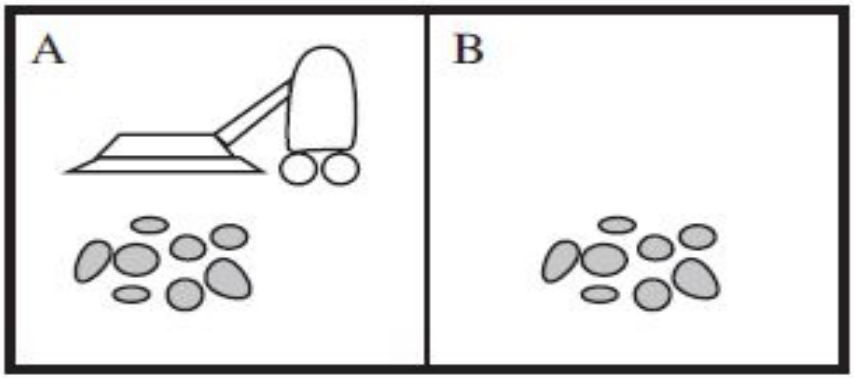
Intelligent Agents



Agent Function

- *Agent function can be tabulated that describes any given agent; for most* agents, this would be a very large table—infinite, in fact, unless we place a bound on the length of percept sequences we want to consider.
- Given an agent to experiment with, we can,in principle, construct this table by trying out all possible percept sequences and recording which actions the agent does in response.
- The table is, of course, an *external characterization* of the agent.

- Internally, the agent function for an artificial agent will be implemented by an **agent program.** It is important to keep these two ideas distinct. The agent function is an abstract mathematical description; the agent program is a concrete implementation, running within some physical system.
- 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. It can choose to move left, move right, suck up the dirt, or do nothing.



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*
⋮	⋮

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

```
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-state vacuum environment. This program implements the agent function tabulated in

The Concept of Rationality

- A rational agent is one that does the right thing. 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.
- There is not one fixed performance measure for all tasks and agents; typically, a designer will devise one appropriate to the circumstances.
- *As a general rule, it is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave.*

- The rationality of an agent is measured by its performance measure.
- Rationality can be judged on the basis of following points:
  - Performance measure which defines the success criterion.
  - Agent prior knowledge of its environment.
  - Best possible actions that an agent can perform.
  - The agents percepts sequence to date
- *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.*

Vacuum Cleaner agent

- Consider the simple vacuum-cleaner agent that cleans a square if it is dirty and moves to the other square if not; this is the agent function tabulated earlier.
- The performance measure is, what is known about the environment, and what sensors and actuators the agent has. Let us assume the following:
- The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
- The "geography" of the environment is known *a priori but the dirt distribution* and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The Left and Right actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
- The only available actions are Left , Right, and Suck.
- The agent correctly perceives its location and whether that location contains dirt.
- We claim that *under these circumstances the agent is indeed rational; its expected performance* is at least as high as any other agent's.

- same agent would be irrational under different circumstances.
- For example, once all the dirt is cleaned up, the agent will oscillate needlessly back and forth; if the performance measure includes a penalty of one point for each movement left or right, the agent will fare poorly.
- A better agent for this case would do nothing once it is sure that all the squares are clean.
- If clean squares can become dirty again, the agent should occasionally check and re-clean them if needed.
- If the geography of the environment is unknown, the agent will need to explore it rather than stick to squares A and B.
- An omniscient agent knows the *actual outcome of its actions and can act accordingly; but omniscience is* impossible in reality.

Omniscience , Learning and Autonomy

- An omniscient agent knows the *actual outcome of its actions and can act accordingly; but omniscience is* impossible in reality.

Example:

- I am walking along the Champs Elys´ees one day and I see an old friend across the street. There is no traffic nearby and I'm not otherwise engaged, so, being rational, I start to cross the street. Meanwhile, at 33,000 feet, a cargo door falls off a passing airliner, and before I make it to the other side of the street I am flattened. Was I irrational to cross the street? It is unlikely that my obituary would read "Idiot attempts to cross street."

- Rationality is not the same as perfection.
- Rationality maximizes *expected performance, while perfection maximizes actual performance.*
- *Retreating* from a requirement of perfection is not just a question of being fair to agents.
- The point is that if we expect an agent to do what turns out to be the best action after the fact, it will be impossible to design an agent to fulfill this specification.

AGENT

- If an agent does not look both ways before crossing a busy road, then its percept sequence will not tell it that there is a large truck approaching at high speed.
- Does our definition of rationality say that it's now OK to cross the road? Far from it! First, it would not be rational to cross the road given this uninformative percept sequence: the risk of accident from crossing without looking is too great. Second, a rational agent should choose the "looking" action before stepping into the street, because looking helps maximize the expected performance. Doing actions *in order to modify future percepts—sometimes called* ***information gathering—*** is an important part of rationality.

Agent

- Another example of information gathering is provided by the **exploration** that must be undertaken by a vacuum-cleaning agent in an initially unknown environment.
- Rational agent not only to gather information but also to **learn** as much as possible from 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.
- There are extreme cases in which the environment is completely known *a priori.*
- *In such cases, the agent need not perceive or learn; it simply acts correctly.*

Agent

- Agent relies on the prior knowledge of its designer rather than on its own percepts, then it is said that the agent lacks **autonomy.**
- A rational agent should be **autonomous**—it should learn what it can to compensate for partial or incorrect prior knowledge.
- Eg. a vacuum-cleaning agent that learns to foresee where and when additional dirt will appear will do better than one that does not.

Rational Agent

- It would be reasonable to provide an artificial intelligent agent with some initial knowledge as well as an ability to learn.
- After sufficient experience of its environment, the behavior of a rational agent can become effectively *independent of its prior knowledge.*
- *Hence, the* incorporation of learning allows one to design a single rational agent that will succeed in a vast variety of environments.

Nature of Environments

- Task environments, which are essentially the “problems” to which rational agents are the “solutions.”
- We begin by showing how to specify a task environment, illustrating the process with a number of examples.

Specifying the task environment:

- We had to specify the performance measure, the environment, and the agent’s actuators and sensors.
- We group all these under the heading of the task environment.
- For the acronymically minded, we call this the PEAS (Performance, Environment, Actuators, Sensors) description.

THE NATURE OF ENVIRONMENTS

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, keyboard

Figure 2.4 PEAS description of the task environment for an automated taxi.

Taxi driver

- **Performance :** Desirable qualities include getting to the correct destination; minimizing fuel consumption and wear and tear; minimizing the trip time or cost; minimizing violations of traffic laws and disturbances to other drivers; maximizing safety and passenger comfort; maximizing profits.
- **Environment :**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. Obviously, the more restricted the environment, the easier the design problem.

- 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 controllable video cameras so that it can see the road; it might augment these with infrared or sonar sensors to detect distances to other cars and obstacles. To avoid speeding tickets, the taxi should have a speedometer, and to control the vehicle properly, especially on curves, it should have an accelerometer. To determine the mechanical state of the vehicle, it will need the usual array of engine, fuel, and electrical system sensors. Like many human drivers, it might want a global positioning system (GPS) so that it doesn't get lost. Finally, it will need a keyboard or microphone for the passenger to request a destination.

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of 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 of scene categorization	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	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Properties of Environment

- Observability: full *vs.* partial *vs. non*
- Deterministic *vs.* stochastic
- Episodic *vs.* sequential
- Static *vs. … vs.* dynamic
- Discrete *vs.* continuous

Types of Environment

- There are different sorts of environments, which affect what an agent has to be able to cope with. In designing agents, one should always consider the pair of agent and environment together.

- **Fully Observable vs. Partially Observable**: If an agent's sensors give it full access to the complete state of the environment, the environment is fully observable, otherwise it is only partially observable or unobservable.
- A task environment is effectively fully observable if the sensors detect all aspects that are *relevant to the choice of action; relevance, in turn, depends on the* performance measure.
- Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world.
- An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data

Single agent vs. multiagent:

- An agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two agent environment.
- For Example: In chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's performance measure.
- Thus, chess COMPETITIVE is a **competitive multiagent environment.**
- **In the taxi-driving** environment, on the other hand, avoiding collisions maximizes the performance measure of all agents, so it is a partially **cooperative multiagent environment.**
- It is also **partially competitive** because, for example, only one car can occupy a parking space.
- The agent-design problems in multiagent environments are often quite different from those in single-agent environments; for example, **communication** often emerges as a **rational behavior in multiagent** environments.

Deterministic vs. Stochastic:

- If the next state of the environment is completely determined by the current state and the agent's selected action, the environment is deterministic.
- An environment may appear stochastic if it is only partially observable. An agent need not worry about uncertainty in a fully observable, deterministic environment.
- If the environment is partially observable, however, then it could *appear to be* stochastic.
- Most real situations are so complex that it is impossible to keep track of all the unobserved aspects; for practical purposes, they must be treated as stochastic.
- Taxi driving is clearly stochastic in this sense, because one can never predict the behavior of traffic exactly.

Episodic vs. Sequential

- In an episodic task environment, the agent's experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action.
- Crucially, the next episode does not depend on the actions taken in previous episodes. Many classification tasks are episodic.
- If future decisions do not depend on the actions an agent has taken, just the information from its sensors about the state it is in, then the environment is episodic.
- Eg.an agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions;
- In sequential environments, on the other hand, the current decision could affect all future decisions.
- Chess and taxi driving are sequential: in both cases, short-term actions can have long-term consequences.
- Episodic environments are much simpler than sequential environments because the agent does not need to think ahead.

Static vs. Dynamic :

- If the environment can change while the agent is deciding what to do, the environment is dynamic.
- Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time. If the environment itself does not change with the passage of time but the agent's performance score does, then environment is **semidynamic.**
- Taxi driving is **dynamic:** the other cars and the taxi itself keep moving while the driving algorithm dithers about what to do next. Chess, when played with a clock, is semidynamic. Crossword puzzles are static.

Discrete vs. Continuous

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

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
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Examples of task environments and their characteristics.

THE STRUCTURE OF AGENTS

- The job of AI is to design an **agent program** that implements the agent function— the mapping from percepts to actions.

  agent = architecture + program .

- The architecture makes the percepts from the sensors available to the program, runs the program, and feeds the program's action choices to the actuators as they are generated.
- agent programs take the current percept as input from the sensors and return an action to the actuators

```
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.

- It keeps track of the percept sequence and then uses it to index into a table of actions to decide what to do.

The table—an example of which is given for the vacuum world , represent explicitly the agent function that the agent program embodies.

Let P be the set of possible percepts and let T be the lifetime of the agent (the total number of percepts it will receive). The lookup table will contain $\sum_{t=1}^{T} |P|^t$ entries.

Consider the automated taxi: the visual input from a single camera comes in at therate of roughly 27 megabytes per second (30 frames per second, 640×480 pixels with 24bits of color information). This gives a lookup table with over $10^{250,000,000,000}$ entries for an hour's driving.

Even the lookup table for chess—a tiny, well-behaved fragment of the real world—would have at least 10^{150} entries.

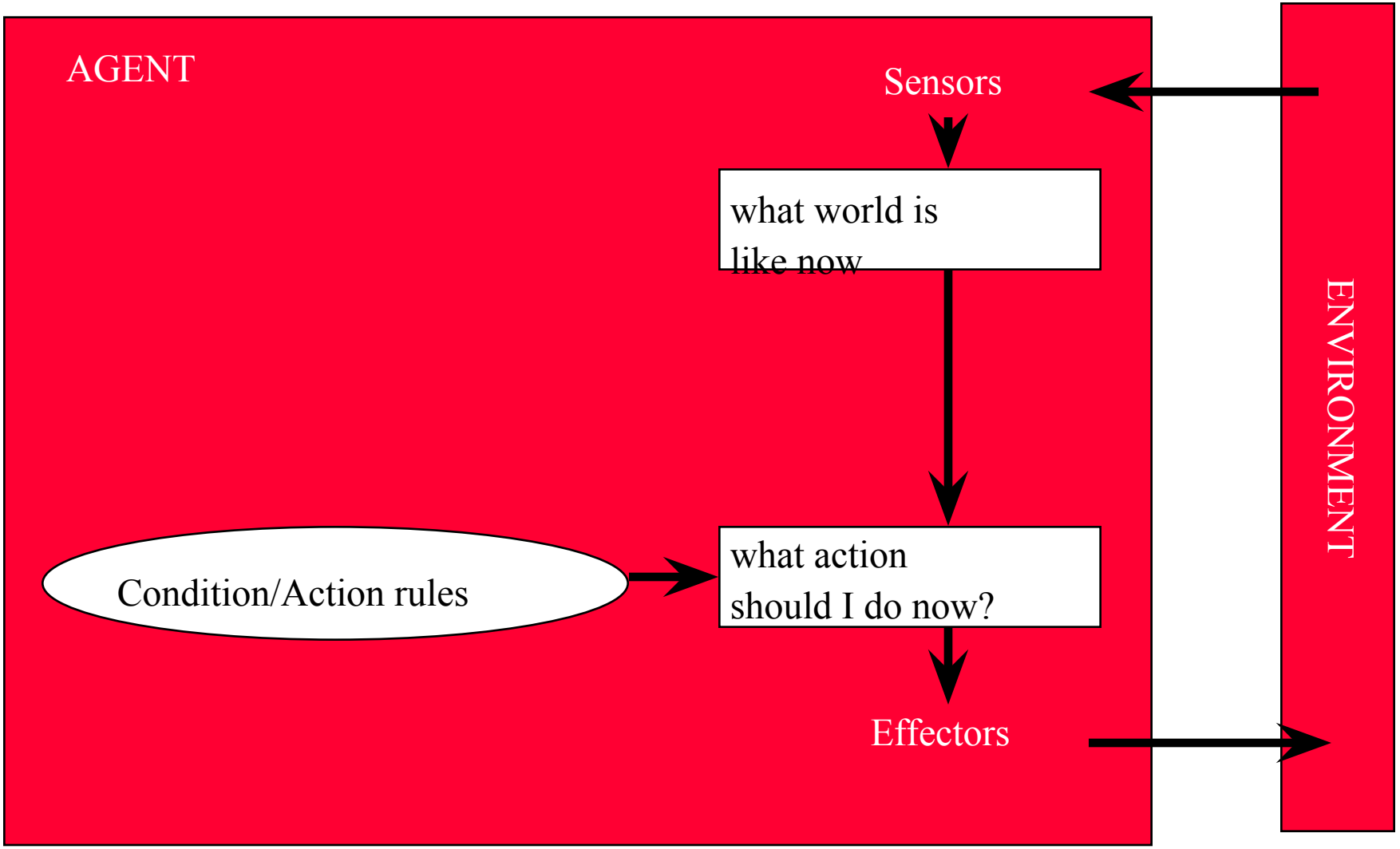
Types of AI Agents

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 given below:

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

Simple reflex agents



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. Such as a Room Cleaner agent, it works only if there is dirt in the room.
- Problems for the simple reflex agent design approach:
  - They have very limited intelligence
  - They do not have knowledge of non-perceptual parts of the current state
  - Mostly too big to generate and to store.
  - Not adaptive to changes in the environment.

Agents function

```
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
```

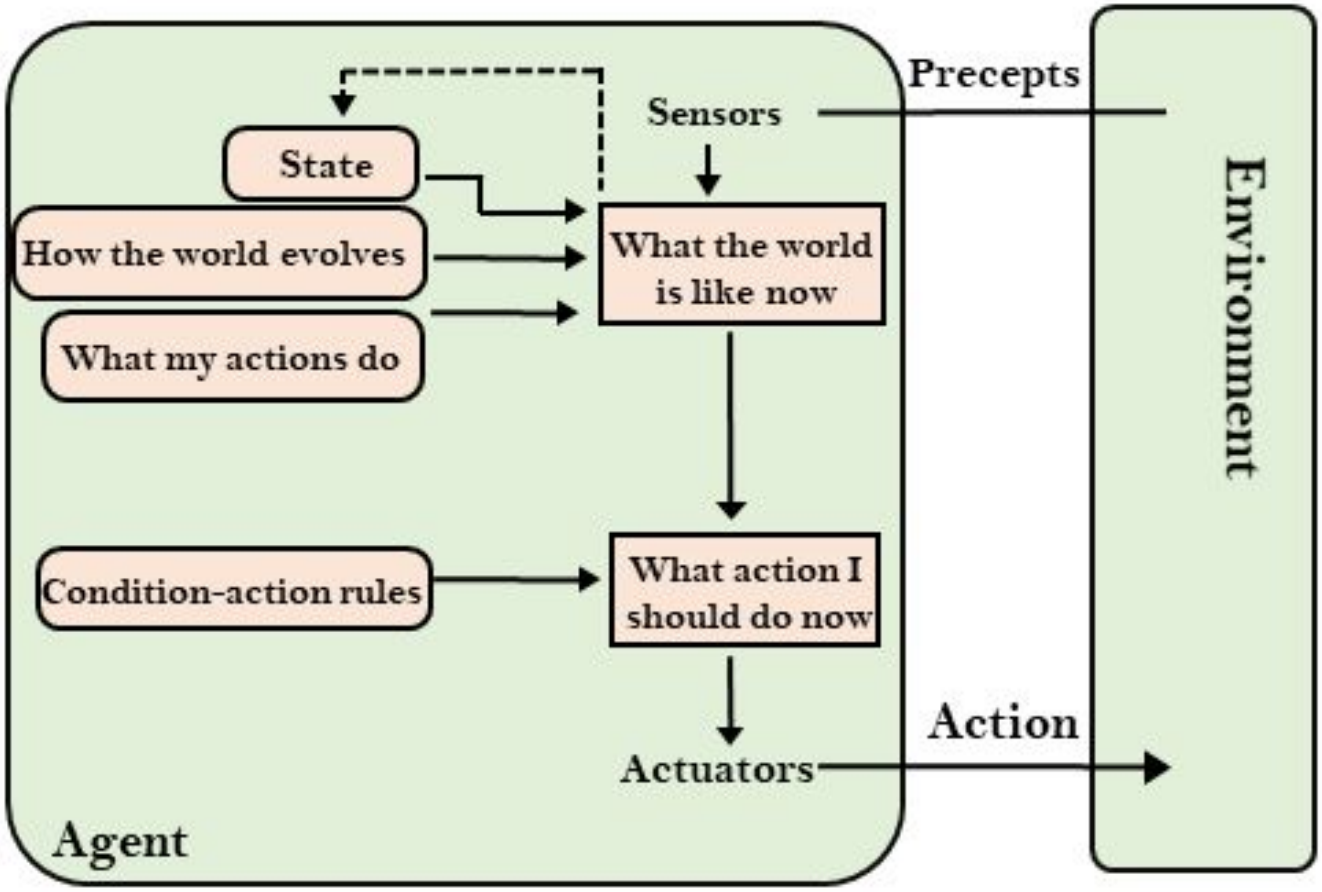
Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

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 independently of the agent- an overtaking car generally will be closer behind than it was a moment ago.
  - How the agent's action affects the world. when the agent turns the steering wheel clockwise, the car turns to the right, or that after driving for five minutes north bound on the freeway, one is usually about five miles north of where one was five minutes ago.

- This knowledge about "how the world works"—whether implemented in simple Boolean circuits or in complete scientific theories—is called a **model of the world. An agent that uses such a** model is called a **model-based agent.**
- Current percept is combined with the old internal state to generate the updated description of the current state, based on the agent's model of how the world works

Model-based reflex agent



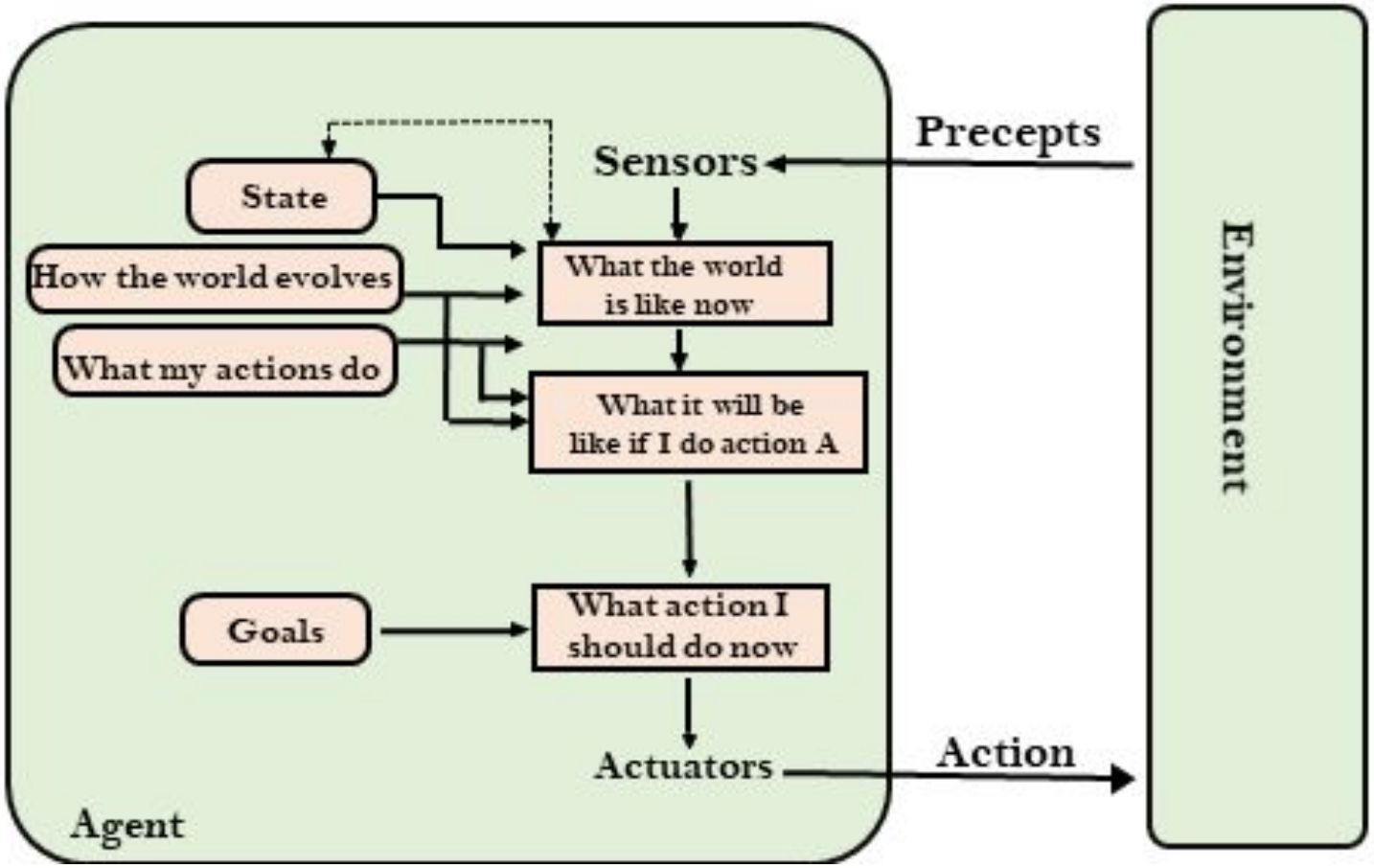
```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, 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

- 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.

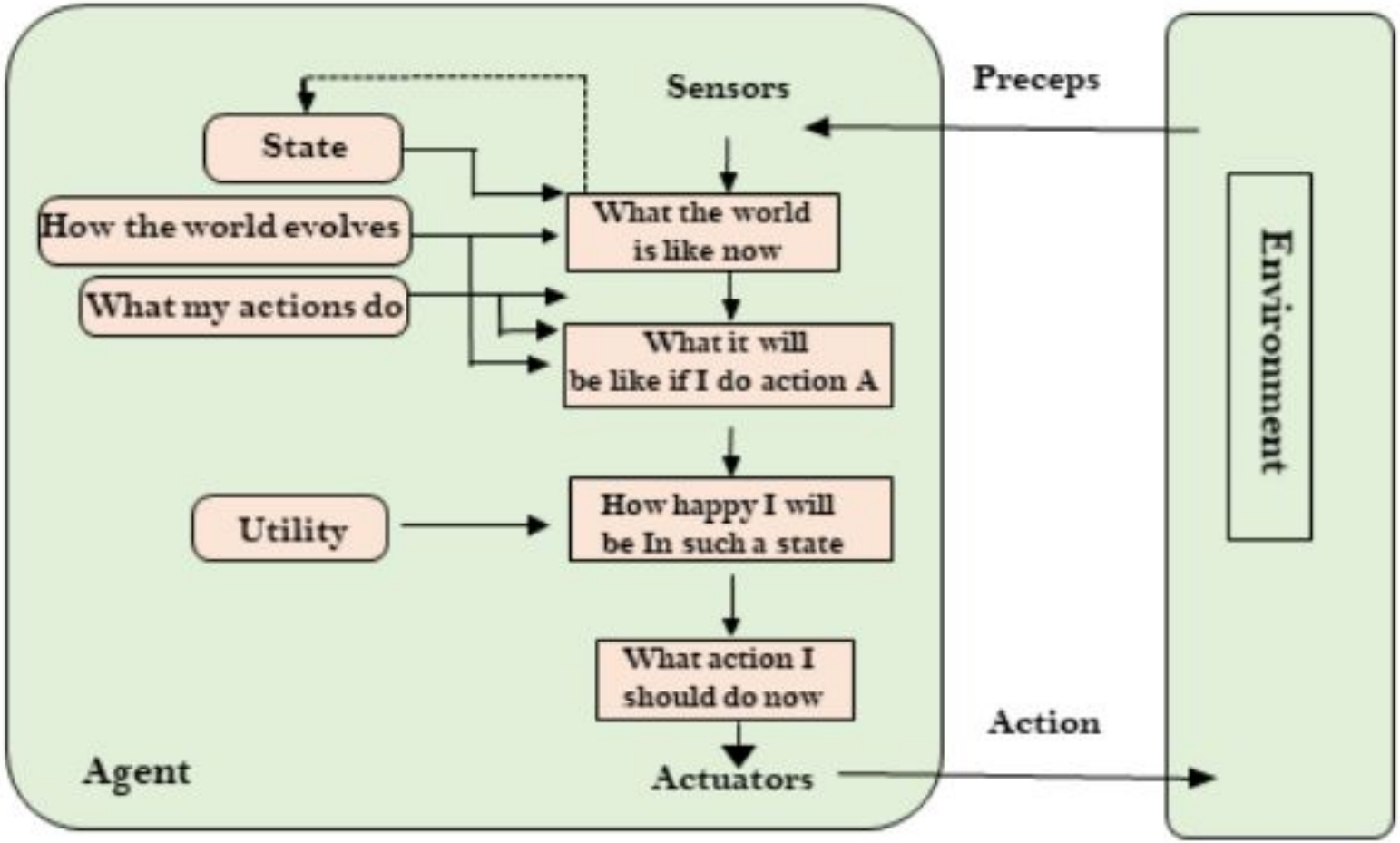
Goal-based agents



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 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.

Utility-based agents



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:
  - **Learning element:** It is responsible for making improvements by learning from environment
  - **Critic:** Learning element takes feedback from critic which describes that how well the agent is doing with respect to a fixed performance standard.
  - **Performance element:** It is responsible for selecting external action
  - **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.

Learning Agents