Intelligent Agents

Unit 2

Artificial Intelligence

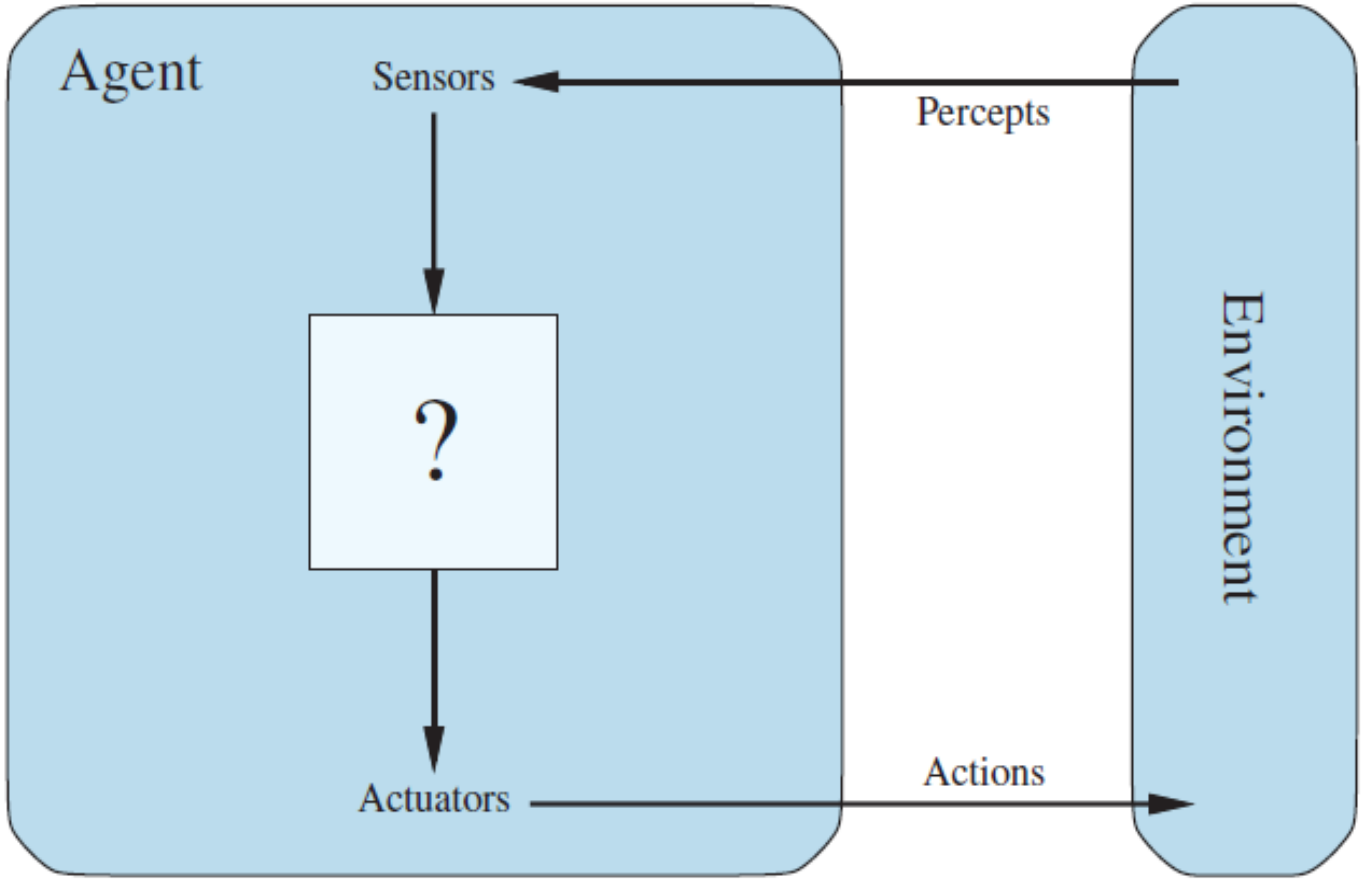
BSc CSIT, 4th Semester

Bibliography

[1] S. J. Russell and P. Norvig. 2020, *Artificial Intelligence: A modern approach, Fourth Edition, Pearson.*

[2] T.B.Shahi, *Artificial Intelligence.B.Sc.CSIT/BIM, KEC Publication.*

Agent



[1]

Agent

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.
- Eg: 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 [1].

Environment

- What is environment?
- The environment could be everything—the entire universe!
- But 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*.

- ***In general, an agent's choice of action at any given instant can depend on its built-in knowledge and on the entire percept sequence observed to date, but not on anything it hasn't perceived [1].***

Rationality

- A rational agent is one that does the right thing.
- But what is the right thing?
- Right action is the one that will cause the agent to be most successful.
- Therefore we need some way to measure success of an agent.
- ***Performance measures*** are the criterion for success of an agent behavior.
- As a general rule, it is better to design performance measures according to what one actually wants to be achieved in the environment, rather than according to how one thinks the agent should behave [1].
- Eg: Vacuum Cleaner : amount of dirt removed vs having a clean floor

Rationality

- What is rational at any given time depends on four things [1]:
- • 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:
- ***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.***

PEAS Description of Agents

- Starting to think about building rational agents [1].
- In designing an agent, the first step must always be to specify the task environment as fully as possible.
- task environments, which are essentially the “problems” to which rational agents are the “solutions.”
- PEAS description of the environment:
- – Performance Measure
- – Environment
- – Actuators
- – Sensors

Automated Taxi Driver Example

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

[1]

Other Examples

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

[1]

Types of task environment

- The range of task environments that might arise in AI is obviously vast.
- We can, however, identify a fairly small number of dimensions along which task environments can be categorized.
- These dimensions determine, to a large extent, the appropriate agent design and the applicability of each of the principal families of techniques for agent implementation [1].
- Fully observable vs. partially observable
- Single-agent vs. Multiagent
- Deterministic vs. Nondeterministic
- Episodic vs. Sequential
- Static vs. Dynamic
- Discrete vs. Continuous
- Known vs. unknown

Fully observable vs. partially observable

- If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is fully observable.
- A task environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action;
- 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
- For example: a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares, and an automated taxi cannot see what other drivers are thinking.
- If the agent has no sensors at all then the environment is unobservable [1].

Single Agent vs Multiagent

- Single agent acting in an environment vs multiple agents
- Problem in multiagent environment: who to consider another agent?
- The key distinction is whether agent B's behavior is best described as maximizing a performance measure whose value depends on agent A’s behavior.
- 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 [1].
- If the interest of one agent to maximize its performance measure:
- Hampers another agents performance measure: Competitive multiagent
- Aids another agents performance measure: Cooperative multiagent

Deterministic vs. Nondeterministic

- If the next state of the environment is completely determined by the current state and the action executed by the agent(s), then we say the environment is deterministic; otherwise, it is nondeterministic.
- In principle, there is no uncertainty in a fully observable, deterministic environment.
- a model of the environment is ***stochastic*** if it explicitly deals with probabilities (e.g., “there’s a 25% chance of rain tomorrow”) and “nondeterministic” if the possibilities are listed without being quantified (e.g., “there’s a chance of rain tomorrow”) [1].

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.
- Furthermore, the next episode does not depend on the actions taken in previous episodes.
- Eg: Many classification tasks are episodic. Detecting defective parts
- On the other hand, in sequential environments, the current decision could affect all future decisions.
- Chess and taxi driving are sequential.
- And in both cases, short-term actions can have long-term consequences [1].

Static vs Dynamic

- If the environment can change while an agent is deliberating, the environment is dynamic for that agent; otherwise, it is static.
- Static environments are easier to deal with because the agent neither have to keep looking at the world while it is deciding on an action, nor it needs to worry about the passage of time.
- While dynamic environments are continuously asking the agent what it wants to do; if it hasn't decided yet, that counts as deciding to do nothing.
- If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is semi-dynamic.
- Taxi driving is clearly dynamic: the other cars and the taxi itself keep moving while the driving algorithm decides about what to do next. Chess, when played with a clock, is semidynamic. Crossword puzzles are static [1].

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.
- For example, 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.).
- Ideally, input from digital cameras is discrete, but is typically treated as representing continuously varying intensities and locations [1].

Known vs Unknown

- Strictly speaking, this distinction refers not to the environment Unknown itself but to the agent's (or designer's) state of knowledge about the "laws of physics" of the environment.
- In a known environment, the outcomes (or outcome probabilities if the environment is nondeterministic) for all actions are given
- Obviously, if the environment is unknown, the agent will have to learn how it works in order to make good decisions.
- It is quite possible for a known environment to be partially observable—for example, in solitaire card games, I know the rules but am still unable to see the cards that have not yet been turned over.
- Conversely, an unknown environment can be fully observable—in a new video game, the screen may show the entire game state but I still don't know what the buttons do until I try them.

Example of task environments

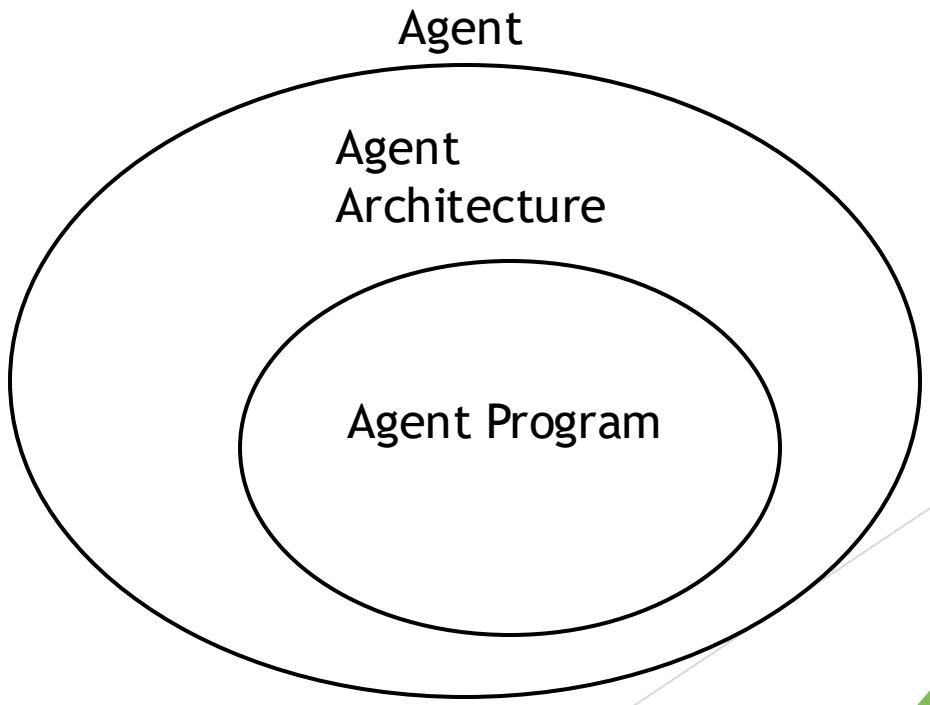
- The hardest case is partially observable, multiagent, nondeterministic, sequential, dynamic, continuous, and unknown.

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

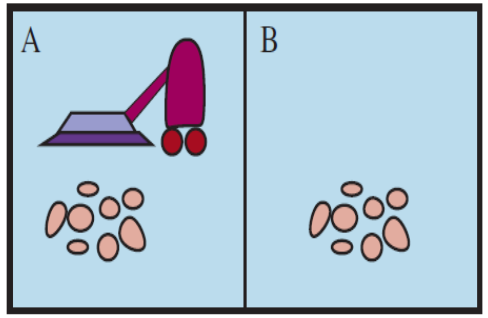
[1]

The Structure of Agents

- ***Agent program*** : that implements the agent function— the mapping from percepts to actions [1].
- ***Agent Architecture***: some sort of computing device with physical sensors and actuators that runs agent program.
- Need harmony between agent architecture and program.



Agent Programs: (TABLE- DRIVEN-AGENT)



[1] Vacuum Cleaner world

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
⋮	⋮

[1] Partial tabulation of a simple agent function

```
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
```

[1] Table-Driven-Agent program pseudocode

Agent Programs: (Problems in TABLE-DRIVEN- AGENT)

- Let P be the set of possible percepts
- T be the lifetime of the agent (the total number of percepts it will receive)
- Then, the lookup table will contain $\sum_{t=1}^{T} |\mathcal{P}|^t$ entries
- means that
- (a) no physical agent in this universe will have the space to store the table
- (b) the designer would not have time to create the table; and
- (c) no agent could ever learn all the right table entries from its experience
- ***The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behavior from a smallish program rather than from a vast table [1].***

Types of Agent (Programs)

- We characterize four basic kinds of agent programs that incorporate the principles underlying almost all intelligent systems:
- Simple reflex agents;
- Model-based reflex agents;
- Goal-based agents; and
- Utility-based agents.

Simple Reflex Agent

- Simplest type of agent
- These agents select actions on the basis of the current percept, ignoring the rest of the percept history
- For example: vacuum agent with agent function tabulated previously
- Its decision is based only on the current location and on whether that location contains dirt.

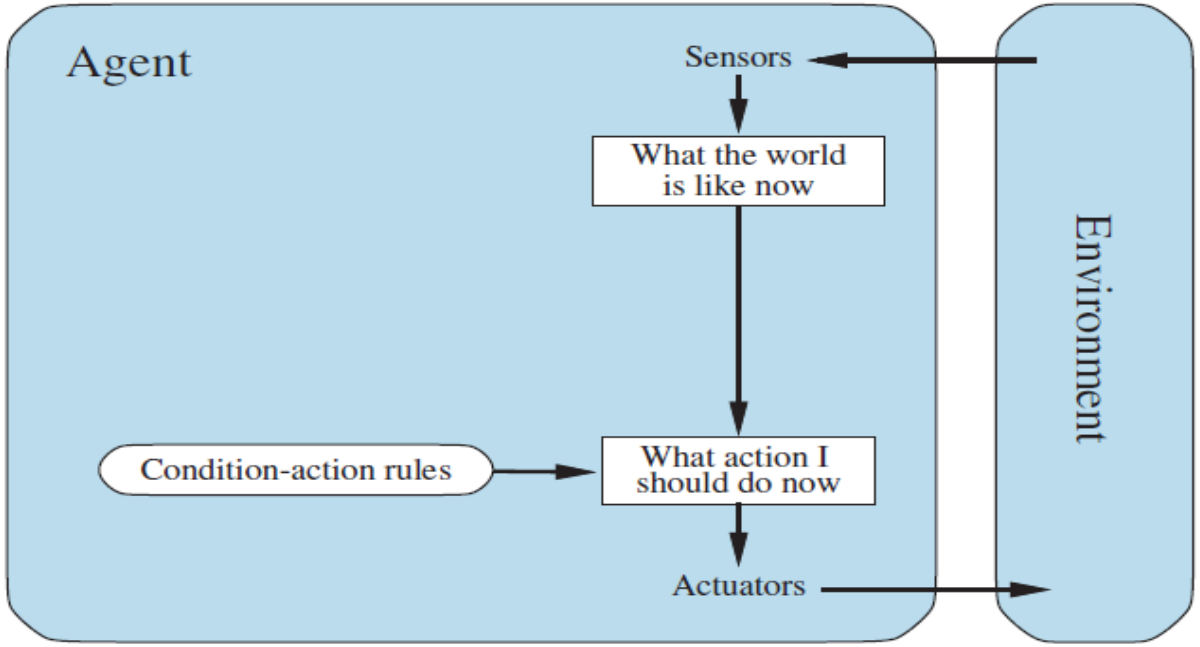
```
function REFLEX-VACUUM-AGENT([location,status]) returns an action

  if status = Dirty then return Suck
  else if location = A then return Right
  else if location = B then return Left
```

[1]

- vacuum agent program is very small compared to the respective table
- Due to ignoring the percept history and focusing on only required percepts at any time
- simple enough to be implemented as a Boolean circuit

Schematic Diagram

- The program in previous slide is specific to one particular vacuum environment.
- A more general and flexible approach is to first develop a general-purpose interpreter for condition–action rules and then to create rule sets for specific task environments. Eg: thermometer and fever



[1]

Agent Program

- INTERPRET-INPUT function generates an abstracted description of the current state from the percept
- RULE-MATCH function returns the first rule in the set of rules that matches the given state description

```
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
```

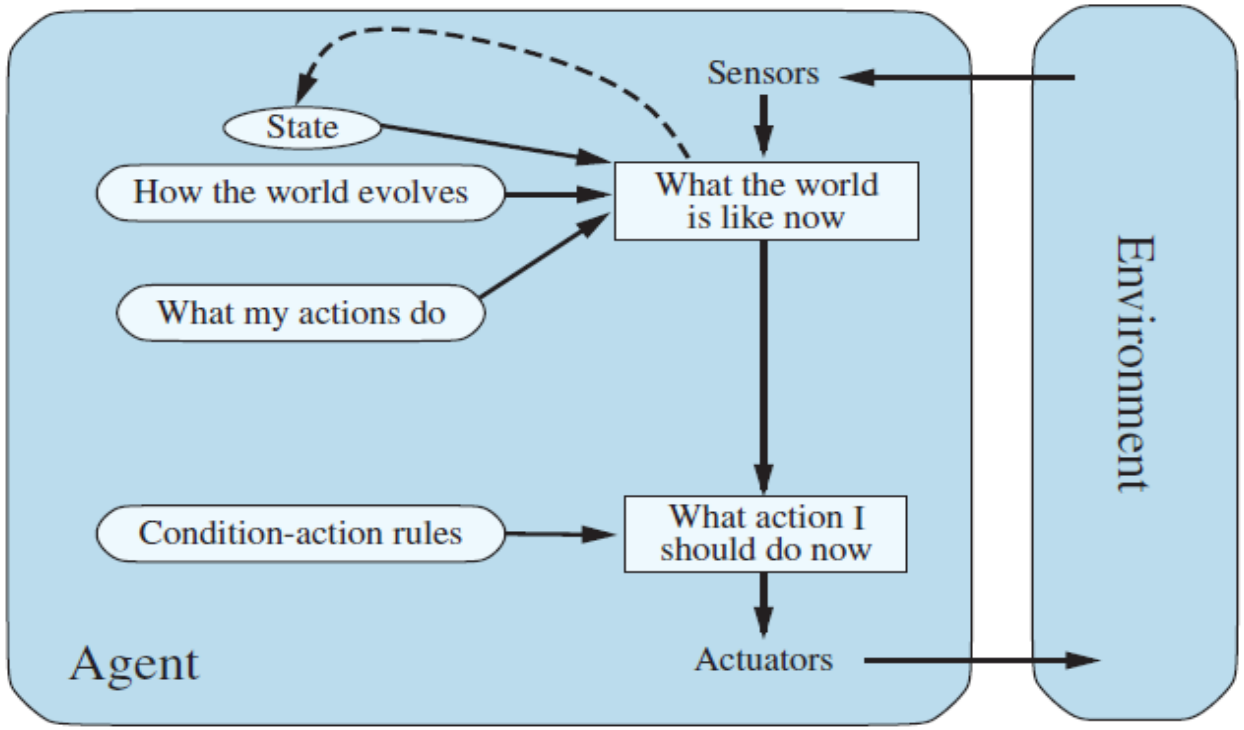
[1]

- ***will work*** only if the correct decision can be made on the basis of just the current percept, i.e., ***only if the environment is fully observable***
- Infinite loops are often unavoidable for simple reflex agents operating in partially observable environments. Eg: vacuum agent without location sensor
- Solution to it would be the agent randomizing its actions but in most cases more sophisticated deterministic agents are better.

Model-based reflex agents

- The most efficient way of handling partial observability by the agent is ***to keep track of the part of the world it can't see now [1].***
- i.e., the agent should maintain some sort of ***internal state*** that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state. Eg: while driving : braking, other cars, keys

[1]



Model based Agent

- To update this internal state information with time two kinds of knowledge needs to be encoded in the agent program
- First one is, some information about ***how the world changes over time***, which includes : ***the effects of the agent's actions*** and ***how the world evolves independently of the agent***.
- Eg: turning steering wheel clockwise results in car moving in right direction, when it's raining the car's cameras can get wet
- This knowledge about "***how the world works***"—whether implemented in simple Boolean circuits or in complete scientific theories—is called a ***transition model*** of the world.
- Secondly, some information about ***how the state of the world is reflected in the agent's percepts*** is needed.
- For example, when the car in front initiates braking, red back lights appear in the forward-facing camera image, and, when the camera gets wet, droplet-shaped objects appear in the image partially obscuring the road. This kind of knowledge is called a ***sensor model***.
- An agent that uses transition model and sensor model together to keep track of the state of the world is called as ***model based agent*** [1].

Model based Agent Program

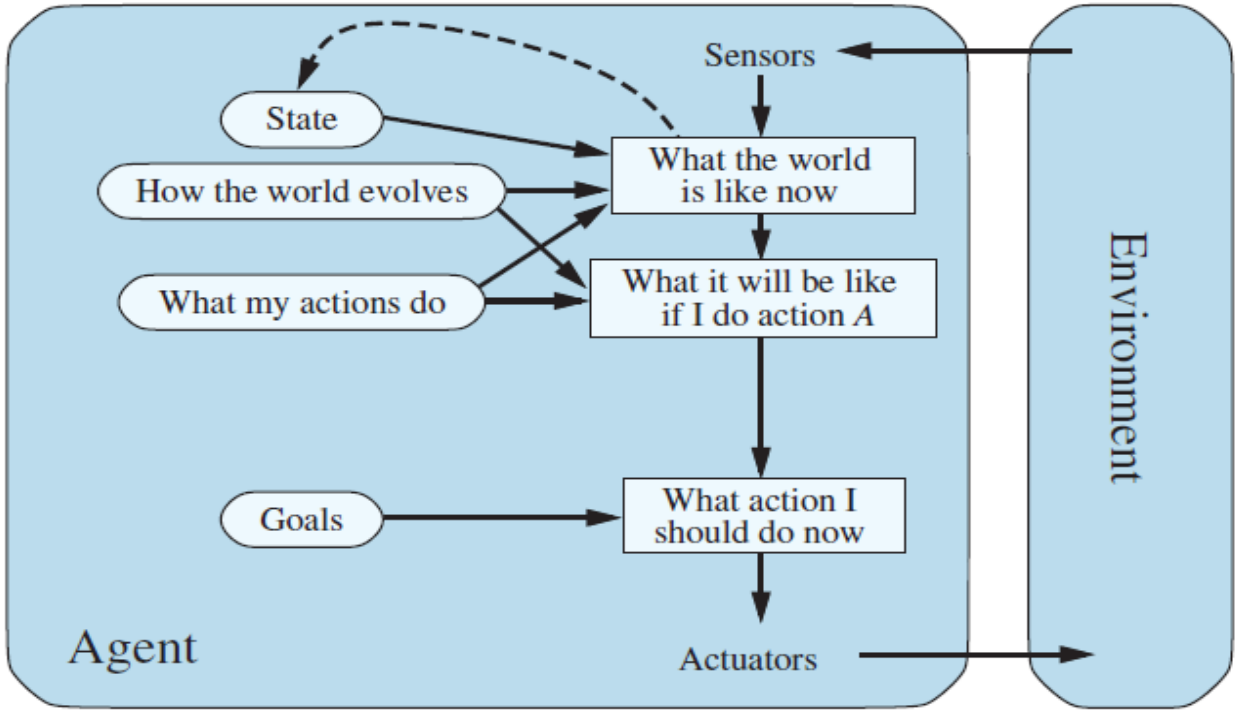
```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              transition_model, a description of how the next state depends on
                    the current state and action
              sensor_model, a description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

  state ← UPDATE-STATE(state, action, percept, transition_model, sensor_model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

[1]

- the function UPDATE-STATE is responsible for creating the new internal state description. The details of how models and states are represented vary widely depending on the type of environment and the particular technology used in the agent design.
- Not possible to be "***exact***" for partially observable environment.
- Best guess(es) for “what the world is like now”. Eg: large truck in front of a car
- ***Thus, uncertainty about the current state may be unavoidable, but the agent still has to make a decision.***

Goal based Agents

- Knowing something about the current state of the environment is not always enough to decide what to do. Eg: road junction
- Thus, **Goal based Agents** combine **goal** information, that describes situations that are desirable, with the model (the same information as was used in the model-based reflex agent), to choose actions that (eventually) achieve the goal [1].



[1]

Goal based Agents

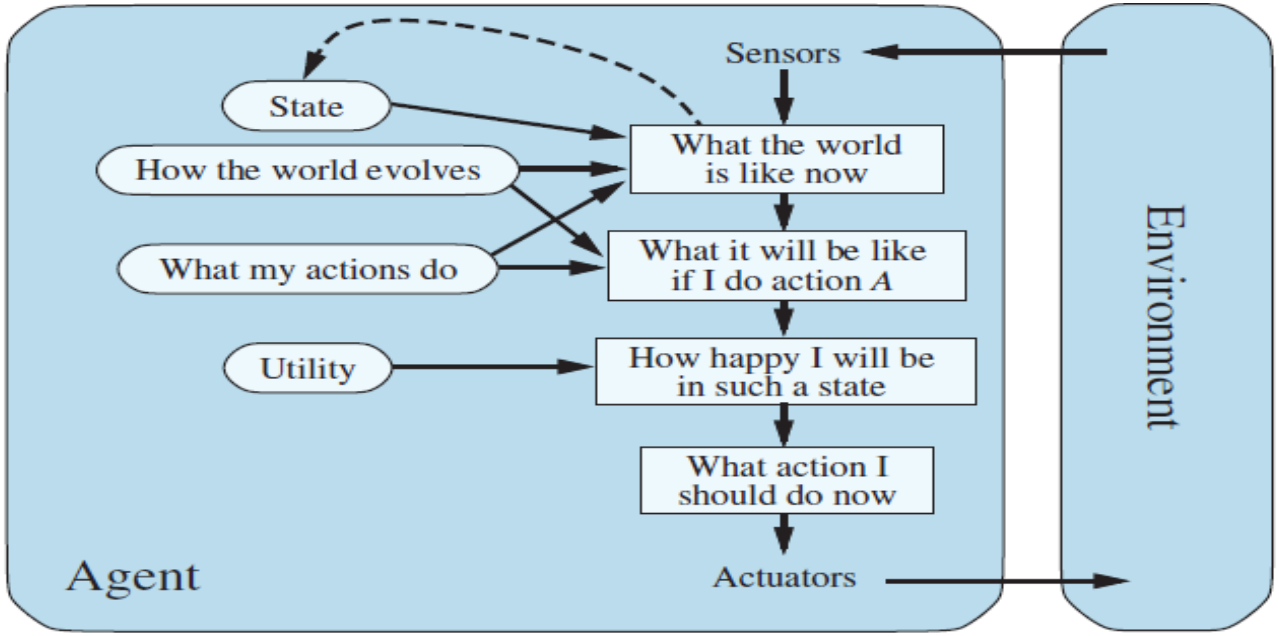
- Selection of goal-based action sequences is sometimes straightforward –for example, when goal satisfaction results immediately from a single action,
- and sometimes it will be more tricky—for example, when the agent has to consider long sequences of twists and turns in order to find a way to achieve the goal.
- ***Search*** in next unit is a subfield of AI addressing this issue [1].

Goal based Agents Vs Reflex Agents

- Decision making here is fundamentally different from the condition– action rules because it involves consideration of the future.
- In the reflex agent designs, this information is not explicitly represented, because the built-in rules map directly from percepts to actions.
- The **reflex agent** brakes when it sees brake lights, period. It has no idea why. A **goal-based agent** brakes when it sees brake lights because that's the only action that it predicts will achieve its goal of not hitting other cars.
- Although the goal-based agent appears less efficient, it is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified.
- For example, a goal-based agent's behavior can easily be changed to go to a different destination by simply designating that destination as the goal.
- The reflex agents rules needs to be replaced for different goal [1].

Utility-based Agents

- Goals alone are not enough to generate high-quality behavior in most environments. Eg: taxi : quicker, safer, more reliable, or cheaper than others.
- A model-based, utility-based agent 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 the average utility that the agent expects to derive given the probabilities and utilities of each outcome [1].



[1]

Utility Function

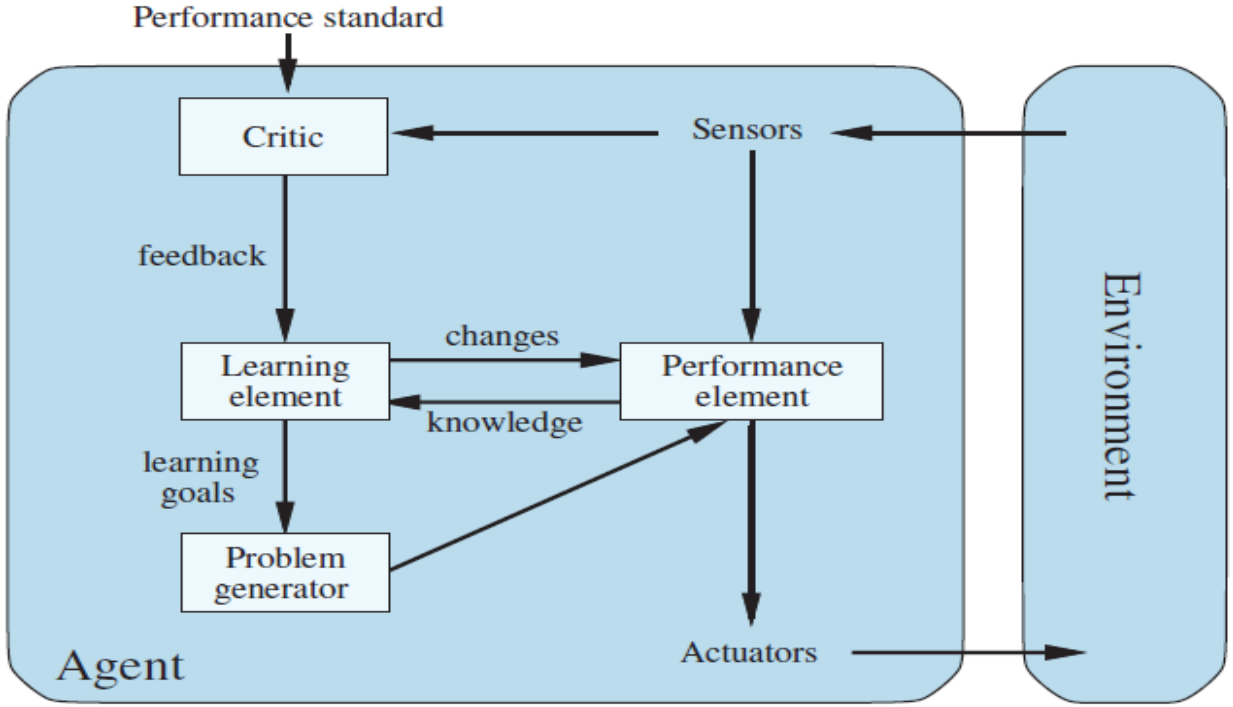
- ***Utility*** is the quality of being useful.
- A performance measure assigns a score to any given sequence of environment states, so it can easily distinguish between more and less desirable ways of getting to the desired states.
- An agent's ***utility function*** is essentially an internalization of the performance measure.
- Provided that the internal utility function and the external performance measure are in agreement, an agent that chooses actions to maximize its utility will be rational according to the external performance measure [1].

Utility based agents vs Goal based agents

- Like goal-based agents, a utility-based agent has many advantages in terms of flexibility and learning in comparison to reflex agents.
- Whereas in two kinds of cases, goals are insufficient but a utility-based agent can still make rational decisions.
- First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function defines the appropriate tradeoff.
- Second, when there are several uncertain goals that the agent can aim for, utility provides a way in which the likelihood of success can be weighed against the importance of the goals [1].
- ***Partial observability and nondeterminism are abundant in the real world, and hence so is decision making under uncertainty. That is why, a rational utility-based agent chooses the action that maximizes the expected utility of the action outcomes.***

Learning Agents

- All the agents discussed earlier can be built as learning agents that can improve the performance of their components so as to generate better actions.
- This was first proposed by Alan Turing in 1950. The method he proposed 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 [1].



[1]

Conceptual components of Learning Agents

- A learning agent can be divided into four conceptual components: Learning element, performance element, critic, and problem generator.
- The most important distinction is between the learning element, which is responsible for making improvements, and 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.
- When trying to design an agent that learns a certain capability, the first question is not "How am I going to get it to learn this?" but *"What kind of performance element will my agent use to do this once it has learned how?"*
- Given a design for the performance element, learning mechanisms can be constructed to improve every part of the agent [1].

Conceptual Components

- The ***critic*** tells the learning element how well the agent is doing with respect to a ***fixed*** performance standard.
- Careful that the agent must not modify performance standard to fit its own behavior.
- In a sense, the performance standard distinguishes part of the incoming percept as a reward (or penalty) that provides direct feedback on the quality of the agent's behavior.
- The ***problem generator*** is responsible for suggesting actions that will lead to new and informative experiences.
- If the performance element had its way, it would keep doing the actions that are best, given what it knows, but if the agent is willing to explore a little and do some perhaps suboptimal actions in the short run, it might discover much better actions for the long run.
- The problem generator's job is to suggest these exploratory actions. This is what scientists do when they carry out experiments [1].

Properties of Intelligent Agents

- The properties of intelligent agents can be classified into internal and external [2].
- Internal characteristics
- ***Learning/reasoning*** : An agent has the ability to learn from previous experience and successfully adapt its own behavior to the environment.
- ***Reactivity***: An agent must to able to appropriately react to influences or information from its environment
- ***Autonomy***: If the agent acts based on the prior knowledge of designer than its own percept, then it can be said that it lacks autonomy. An agent must have control over both its actions and internal state.
- ***Goal-oriented***: An agent should have well defined goals and should be capable to gradually manipulate its environment so as to achieve its own goals [2].

Properties of Intelligent Agents

- ***Communication***: An agent often requires an interaction with its environment to fulfill its tasks, such as human, other agents, and arbitrary information sources.
- ***Cooperation***: Cooperation of several agents permits faster and better solutions for complex tasks that exceed the capabilities of a single agent.
- ***Mobility***: An agent should be able to navigate within the region having any mode of communication networks.
- ***Character***: An agent may demonstrate an external behavior with the character it is trying to impersonate. Eg: humans: humanoids or other animals (birds, insects) [2].