Unit II: Intelligent Agents

Artificial Intelligence

BScCSIT 4th Semester

Lecturer: Mukesh Prasad Chaudhary

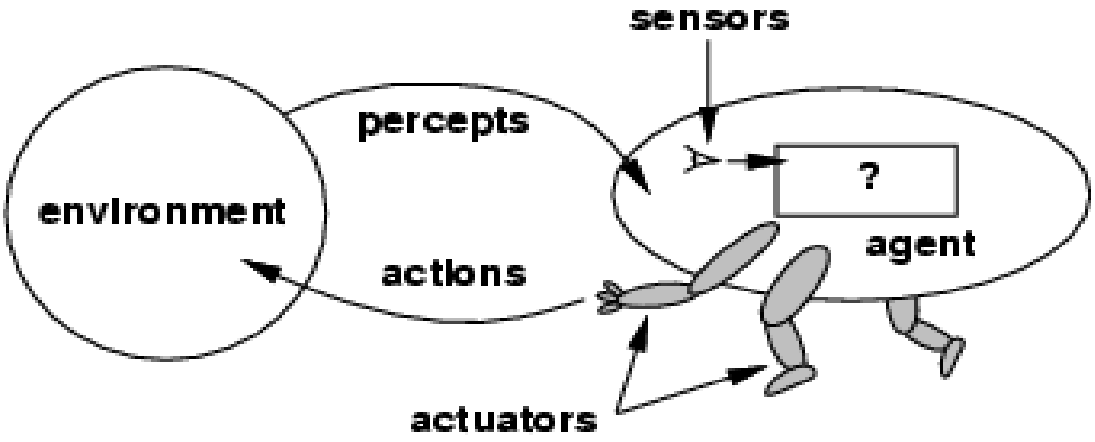
Agents:

- 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
- **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.

Properties of Agents

- **Successful** : An agent is successful to the extent that it accomplishes the specified task in the given environment.
- **Capable** : An agent is capable if it possesses the effectors needed to accomplish the task.
- **Perceptive** : An agent is perceptive if it can distinguish salient characteristics of the world that would allow it to use its effectors to achieve the task.
- **Reactive** : An agent is reactive if it is able to respond sufficiently quickly to events in the world to allow it to be successful.
- **Reflexive** : An agent is reflexive if it behaves in a stimulus-response fashion
- **Interpretive** : An agent is interpretive if can correctly interpret its sensor readings.
- **Rational** : An agent is rational if it chooses to perform commands that it predicts will achieve its goals.
- **Sound** : An agent is sound if it is predictive, interpretive and rational.

Agents and environments



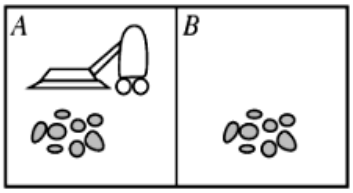
Agent function: A function that specifies the agent's action in response to every possible percept sequence history. Agent function maps perceptions into actions i.e. [*f*: P* → A]

Agent = architecture + program

Architecture: Some sort of computing device with physical sensors and actuators (PC, robotic car).

Agent Program: Takes the current percept as input from the sensors, which is only the available input from the environment, and return an action to the actuators. The agent need to remember the whole percept sequence, if it needs it.

Vacuum-cleaner world



- Two locations: A and B
- Percepts: location and contents, e.g., [A,Dirty]
- Actions: *Left*, *Right*, *Suck*, *NoOp*

Percept sequence	Actions
[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,Clean]	Suck

One simple function is :

if the current square is dirty then suck, otherwise move to the other square

Rational agent

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. An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform. The right action is the one that will cause the agent to be most successful.

Rationality is distinct from omniscience (all knowing with infinite knowledge). Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering, exploration).

Autonomous Agent: An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt).It has ability to learn and adapt.

Performance measure vs. Utility function

- A performance measure (typically imposed by the designer) is used to evaluate the behavior of the agent in environment. It tells does agent do what it's supposed to do in the environment. It determine the objective criterion for success of an agent's behavior. E.g., performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.
- A utility function is used by an agent itself to evaluate how desirable states are. Some paths to the goal are better (more efficient) than others –which path is the best.
- Does agent do what it's supposed to do vs. does agent do it in optimal way
- The utility function may not be the same as the performance measure.
- An agent may have no explicit utility function at all, whereas there is always a performance measure

PEAS: Must first specify the setting for intelligent agent design.

Performance – which qualities it should have?

Environment – where it should act?

Actuators – how will it perform actions?

Sensors – how will it perceive environment?

Examples of PEAS:

Automated taxi driver:

- **Performance measure**: Safe, fast, legal, comfortable trip, maximize profits
- **Environment:** Roads, other traffic, pedestrians, customers
- **Actuators:** Steering wheel, accelerator, brake, signal, horn
- **Sensors:** Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard

Agent: Interactive English tutor

- **Performance measure**: Maximize student's score on test
- **Environment:** Set of students
- **Actuators:** Screen display (exercises, suggestions, corrections)
- **Sensors:** Keyboard

Agent: Medical diagnosis system

- **Performance measure**: Healthy patient, minimize costs, lawsuits
- **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

PEAS for a Satellite image analysis system

- **Performance measure**: Correct image categorization
- **Environment:** downlink from orbiting satellite
- **Actuators**: display categorization of scene
- **Sensors:** color pixel arrays

PEAS for a refinery controller

- **Performance measure**: maximize purity, yield, safety
- **Environment:** refinery, operators
- **Actuators**: valves, pumps, heaters, displays
- **Sensors**: temperature, pressure, chemical sensors

Mathematician's theorem-proving assistant

P: good math knowledge, can prove theorems accurately and in minimal steps/time

E: Internet, library

A: display

S: keyboard

Autonomous Mars rover

P: Terrain explored and reported, samples gathered and analyzed

E: Launch vehicle, lander, Mars

A: Wheels/legs, sample collection device, analysis devices, radio transmitter

S: Camera, touch sensors, accelerometers, orientation sensors, wheel/joint encoders, radio receiver

Internet book-shopping agent

P: Obtain requested/interesting books, minimize expenditure

E: Internet

A: Follow link, enter/submit data in fields, display to user

S: Web pages, user requests

Robot soccer player

P: Winning game, goals for/against

E: Field, ball, own team, other team, own body

A: Devices (e.g., legs) for locomotion and kicking

S: Camera, touch sensors, accelerometers, orientation sensors, wheel/joint encoders

Environment types:

- Fully observable vs. partially observable
- Deterministic vs. stochastic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous
- Single agent vs. multi-agent

Fully Observable vs. partially observable:

- An environment is fully observable if an agent's sensors give it access to the complete state of the environment at each point in time. 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
- Examples: vacuum cleaner with local dirt sensor, taxi driver

Deterministic vs. stochastic

- The environment is **deterministic** if the next state of the environment is completely determined by the current state and the action executed by the agent. In principle, an agent need not worry about uncertainty in a fully observable, deterministic environment
- If the environment is partially observable then it could appear to be **stochastic**
- Examples: Vacuum world is deterministic while taxi driver is not.
- If the environment is deterministic except for the actions of other agents, then the environment is strategic.

Episodic vs. sequential

- In **episodic environments**, the agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself. Examples: classification tasks.
- In **sequential environments**, the current decision could affect all future decisions Examples: chess and taxi driver.

Static vs. dynamic

- The environment is unchanged while an agent is deliberating.
- Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on the action or need it worry about the passage of time
- Dynamic environments continuously ask the agent what it wants to do
- The environment is semi-dynamic if the environment itself does not change with the passage of time but the agent's performance score does
- Examples: taxi driving is dynamic, chess when played with a clock is semi-dynamic, crossword puzzles are static

Discrete vs. continuous

- A limited number of distinct clearly defined states, percepts and actions.
- Examples: Chess has finite number of discrete states, and has discrete set of percepts and actions. Taxi driving has continuous states, and actions

Single agent vs. multi-agent

- An agent operating by itself in an environment is single agent
- Examples: Crossword is a single agent while chess is two-agents

- Question: Does an agent A have to treat an object B as an agent or can it be treated as a stochastically behaving object
- Whether B's behaviour is best described by as maximizing a performance measure whose value depends on agent's A behaviour
- Examples: chess is a competitive multiagent environment while taxi driving is a partially cooperative multiagent environment

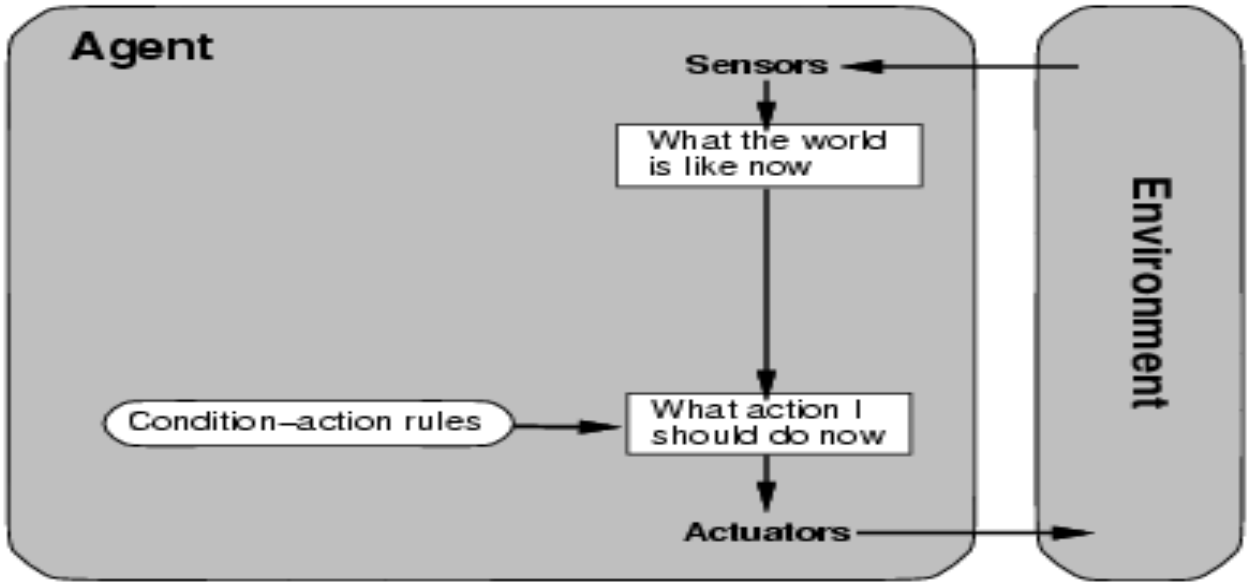
Agent types:

- Four basic types in order of increasing generality:
  - ❖ Simple reflex agents
  - ❖ Model-based reflex agents
  - ❖ Goal-based agents
  - ❖ Utility-based agents

Simple reflex agents

- Select actions based on the current percept ignoring the rest of the percept history.
  Example: simple reflex vacuum cleaner agent

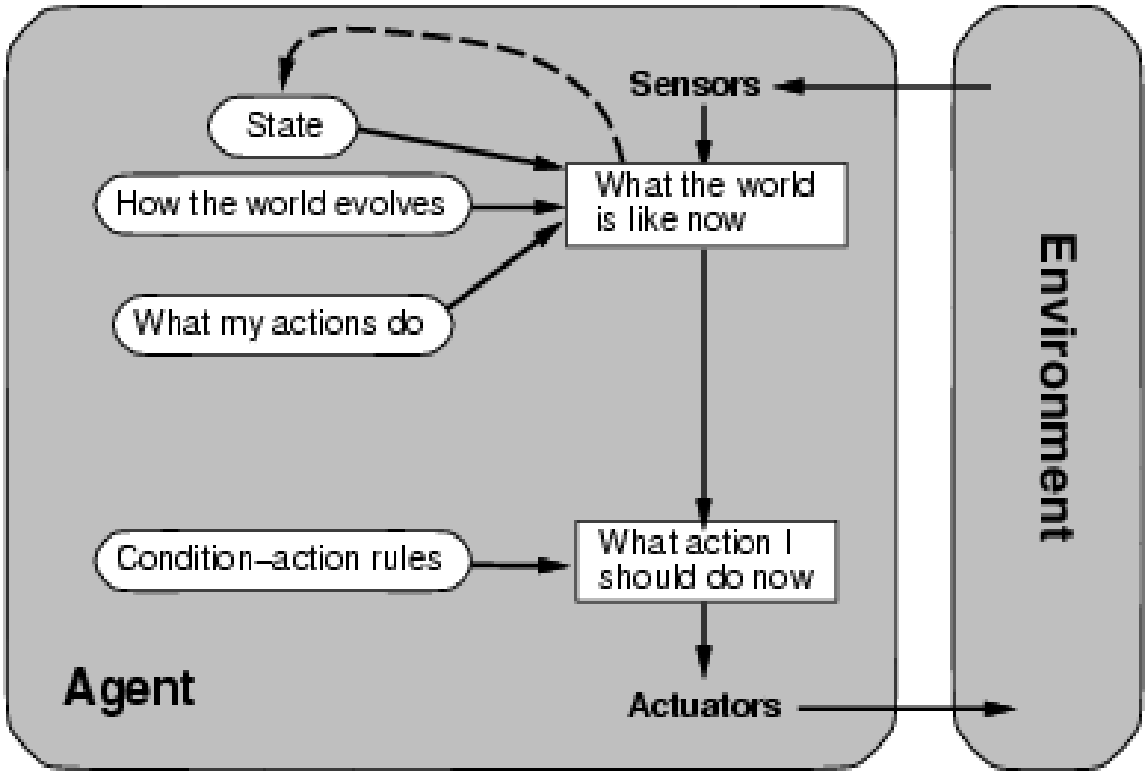
```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```



- function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
  static: *rules,* a set if condition-action rules
  state <-- INTERPRET_INPUT(*percept*)
  rule <-- RULE_MATCH(*state, rules*)
  action <-- RULE_ACTION[*rule*]
  return *action*
- Simple-reflex agents are simple, but they turn out to be of very limited intelligence
- The agent will work only if the correct decision can be made based on the current percept – that is only if the environment is fully observable. Infinite loops are often unavoidable – escape could be possible by randomizing.

Model-based reflex agents

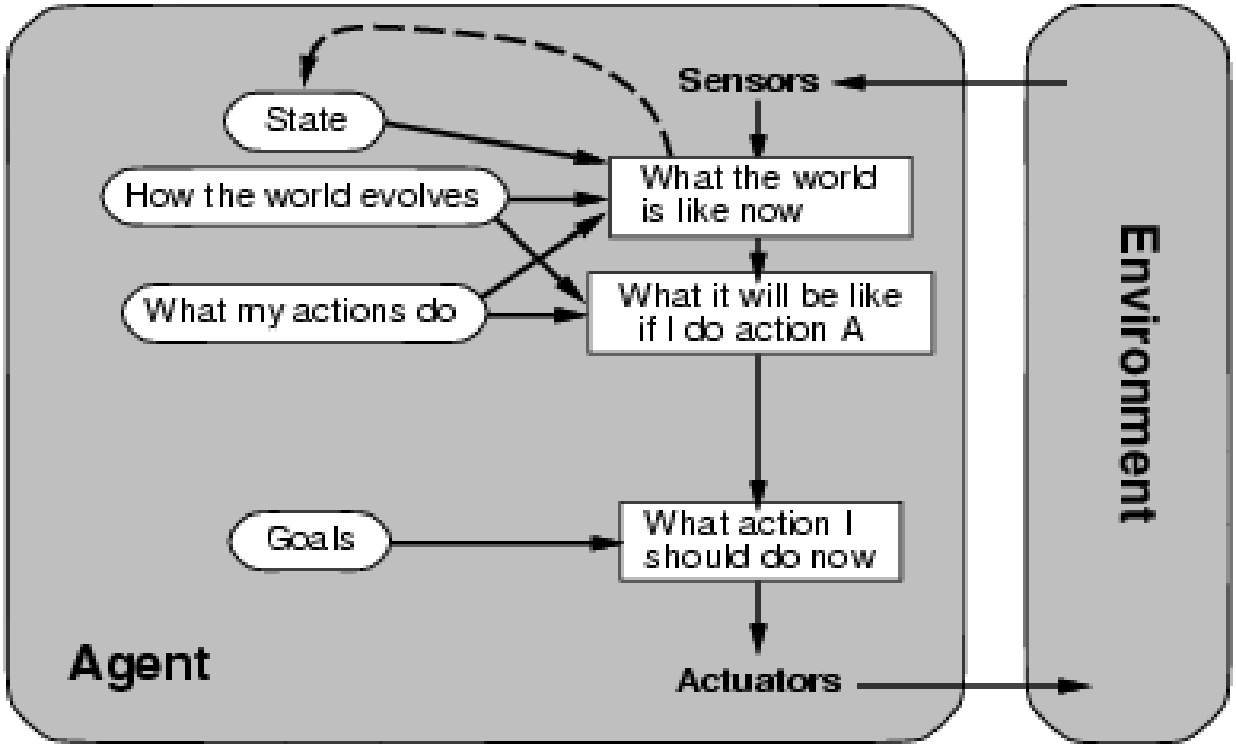
- The agent should keep track of the part of the world it can't see now
- The agent should maintain some sort of internal state that depends on the percept history and reflects at least some of the unobserved aspects of the current state.
- Updating the internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program.
  - ❖ Information about how the world evolves independently of the agent
  - ❖ Information about how the agent's own actions affects the world



- function REFLEX-AGENT-WITH-STATE(*percept*) **returns** an action
  static: *state*, a description of the current world state
  rules, a set of condition-action rules
  action, the most recent action, initially none
  state <-- UPDATE_INPUT(*state, action, percept*)
  rule <-- RULE_MATCH(*state, rules*)
  action <-- RULE_ACTION[*rule*]
  return *action*

Goal-based agents

- Knowing about the current state of the environment is not always enough to decide what to do (e.g. decision at a road junction)
- The agent needs some sort of goal information that describes situations that are desirable. The agent program can combine this with information about the results of possible actions in order to choose actions that achieve the goal. Usually requires search and planning.

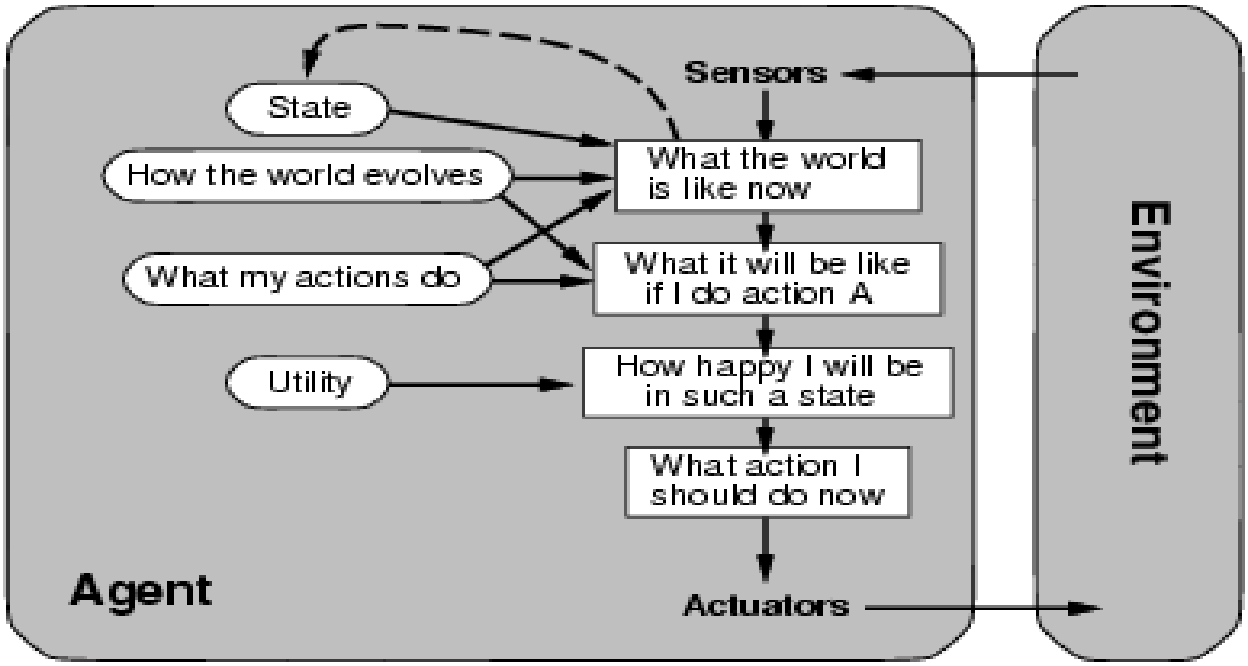


Goal-based agents vs. reflex-based agents

- Although goal-based agents appears less efficient, it is more flexible because the knowledge that supports its decision is represented explicitly and can be modified. On the other hand, for the reflex-agent, we would have to rewrite many condition-action rules.
- The goal-based agent's behavior can easily be changed. The reflex agent's rules must be changed for a new situation.

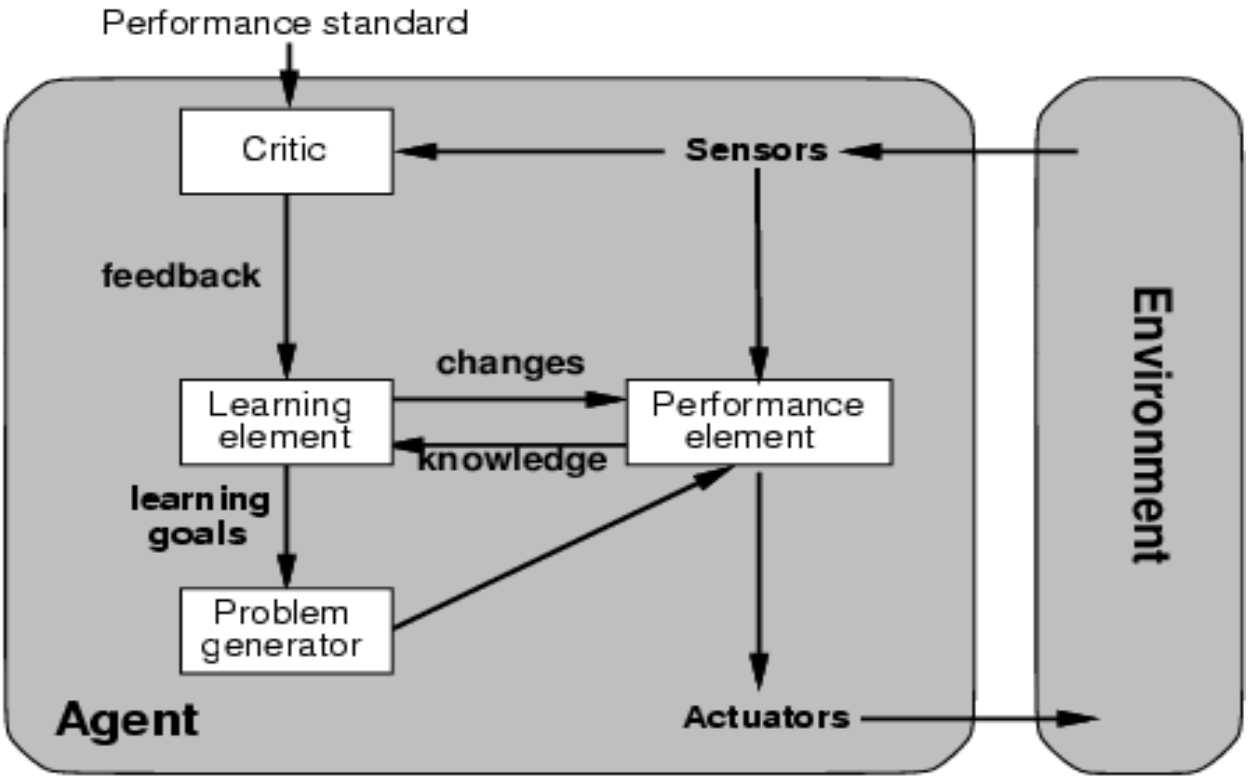
Utility-based agents

- Goals alone are not really enough to generate high quality behavior in most environments – they just provide a binary distinction between happy and unhappy states
- A more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent if they could be achieved
- Happy – Utility (the quality of being useful)
- A utility function maps a state onto a real number which describes the associated degree of happiness



Learning agents

- Turing – instead of actually programming intelligent machines by hand, which is too much work, build learning machines and then teach them.
- Learning also allows the agent to operate in initially unknown environments and to become more competent than its initial knowledge alone might allow.



- Learning element – responsible for making improvements.
- Performance element – responsible for selecting external actions (it is what we had defined as the entire agent before).
- 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.
- Problem generator is responsible for suggesting actions that will lead to a new and informative experiences.

Table-lookup agent

- A trivial agent program: keeps track of the percept sequence and then uses it to index into a table of actions to decide what to do?
- The designers must construct the table that contains the appropriate action for every possible percept sequence
- function TABLE-DRIVEN-AGENT(*percept*) returns an action
  static: *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*

- **Drawbacks:**
    # Huge table (P^T , P: set of possible percepts, T: lifetime)
        - Space to store the table
        - Take a long time to build the table
        - No autonomy
        - Even with learning, need a long time to learn the table entries