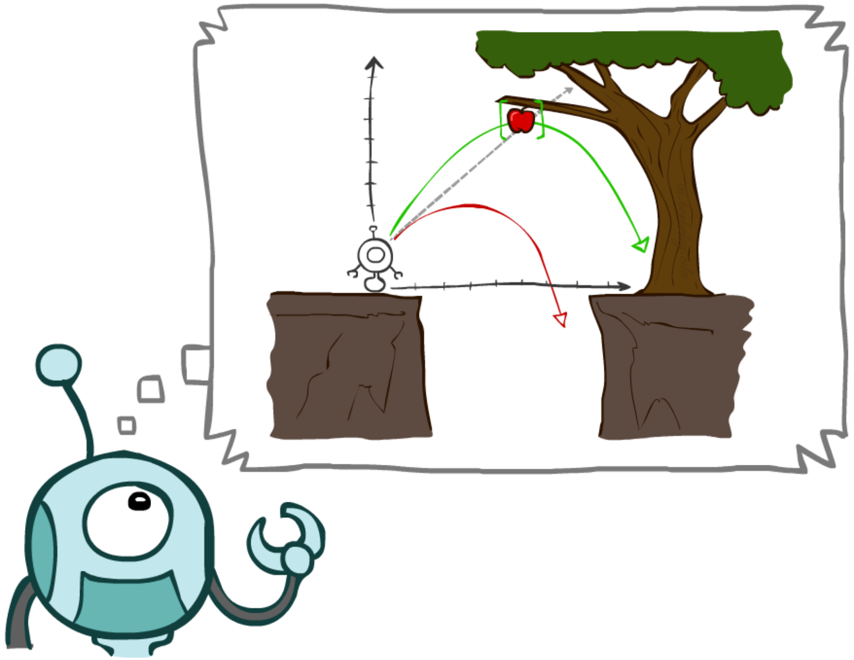
Intelligent Agents

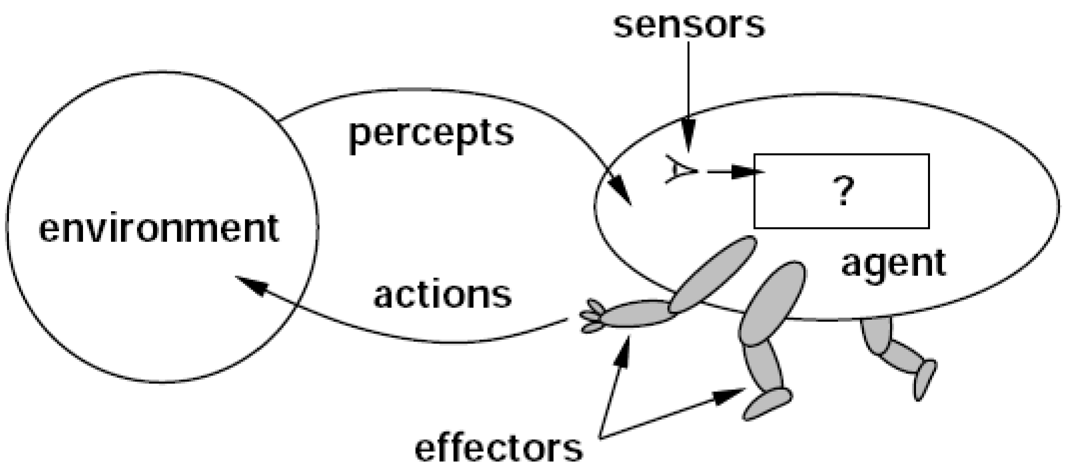


Instructor: Dr. Mohsin Ashraf

Slide credits: AI BERKELEY

What is an Agent?

- *“Intelligent agents continuously perform three functions: **perception** of dynamic conditions in the environment; **action** to affect conditions in the environment; and **reasoning** to interpret perceptions, solve problems, draw inferences, and determine actions.”*



Intelligent Agents

- An intelligent agent may learn from the environment to achieve their goals.
- Following are the main four rules for an AI agent:
  - **Rule 1:** An AI agent must have the ability to perceive the environment.
  - **Rule 2:** The observation must be used to make decisions.
  - **Rule 3:** Decision should result in an action.
  - **Rule 4:** The action taken by an AI agent must be a rational action.

Example of Agents

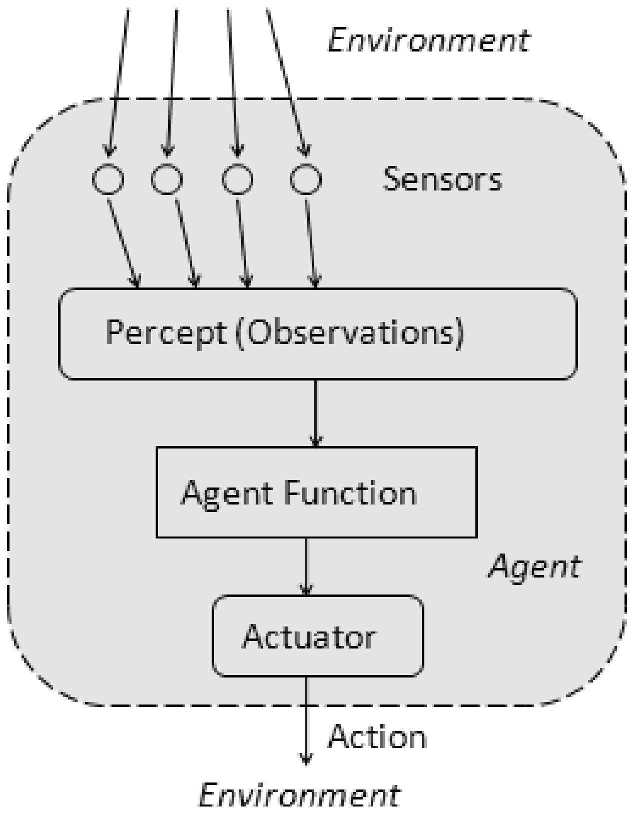
- Human Agent
  - eyes, ears, skin, taste buds, etc. for sensors
  - hands, fingers, legs, mouth, etc. for actuators
- Robot
  - camera, infrared, bumper, etc. for sensors
  - grippers, wheels, lights, speakers, etc. for actuators
- Software Agent
  - functions as sensors
    - information provided as input to functions in the form of encoded bit strings or symbols
  - functions as actuators
    - results deliver the output

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.
- **Rationality** is distinct from **omniscience** (all-knowing with infinite knowledge)
  - **Actual (perfection) vs. Expected**
  - **Best vs. Optimal**

Agent Function and Program

- **Behavior of Agent** – It is the action that agent performs after any given sequence of percepts
- 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
- Problems:
  - what is " the right thing"
  - how do you measure the "best outcome"



Specifying the Task Environment

Performance of Agents

Performance of Agents

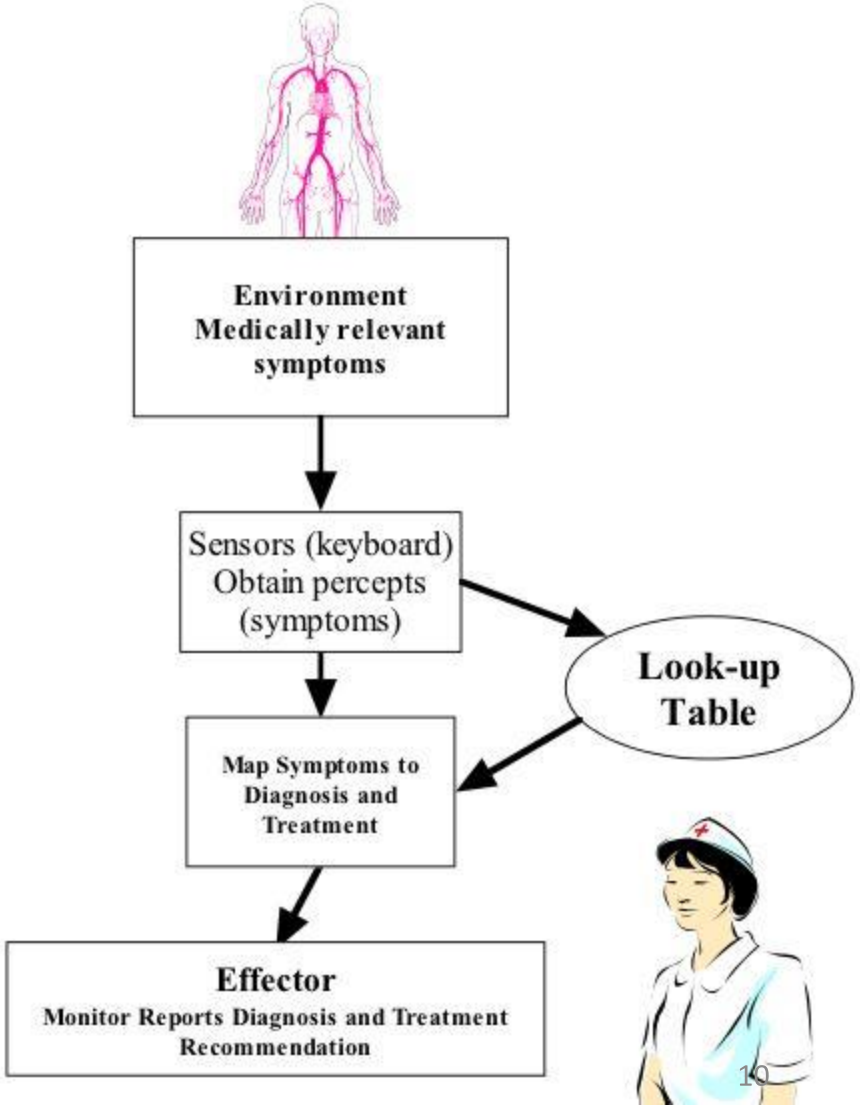
- Behaviour and performance of IA in terms of agent program.
  - Perception to Action Mapping
  - Ideal mapping: specifies which actions an agent ought to take at any point in time.
- Performance measure: a subjective measure to characterize how successful an agent is (e.g., speed, power, accuracy, etc.)
- **PEAS:**
  - **Performance measure,**
  - **Environment,**
  - **Actuators,**
  - **Sensors**

PEAS: Automated Taxi

- Performance measure
  - Correct Destination, Fuel, time, cost, violations, safety, comfort, profit
- Environment
  - rural, urban, other drivers, customers
- Actuators
  - Steering, brake, fuel, display/speaker
- Sensors
  - Camera, radar, accelerometer, engine sensors, microphone

PEAS: 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)



PEAS: 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: Interactive English tutor

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

Environment Types

Properties of environments

vs.

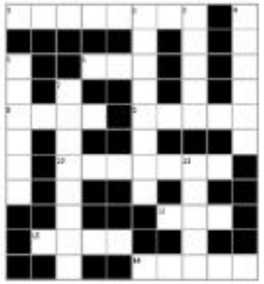
- **Fully observable** vs. **partially observable**
- **Or Accessible vs. inaccessible**
  - Can the agent observe the complete state of the environment?
  - If an agent's sensors gives it access to the complete state of the environment, then we say that environment is fully observable to the agent.
  - A fully observable environment is convenient because the agent need not maintain any internal state to keep track of the world.
  - ***Is the current state of the world fully known at each step?***

Properties of environments

vs.

- **Deterministic** vs. **nondeterministic (Stochastic).**
  - Is there uncertainty in how the world works?
  - If the next state of the environment is completely determined by the current state and the actions selected by the agents, then we say the environment is deterministic.
  - If the environment is inaccessible, then it may *appear* to be nondeterministic (bunch of uncertainties).
  - Taxi driving is clearly stochastic in this sense, because one can never predict the behavior of traffic exactly.

Environment Types



vs.

- **Episodic** vs. **Sequential**:
  - Episodic: next episode doesn’t depend on previous actions.
  - 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.
  - the next episode does not depend on the actions taken in previous episodes
  - For example, an agent that has to spot defective parts on an assembly line (EPISODIC)
  - Chess and Taxi driver (SEQUENTIAL).

Environment Types

- **Static** vs. **Dynamic**:
  - If the environment can change while an agent is agent is performing action, then we say the environment is dynamic for that agent; otherwise, it is static.
  - Dynamic environments are continuously asking the agent what it wants to do
  - Static environments are easy to deal with, because the agent does not keep on looking at the environment while it is deciding on an action.
  - The environment is **semi-dynamic** if the environment itself does not change with the passage of time but the agent's performance score does.

Environment Types

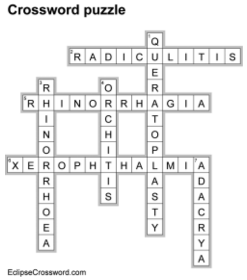
- **Discrete** vs. **continuous**:
  - Is there a finite (or countable) number of possible environment states?
  - The environment is discrete if the number of actions and possible states of the environment is finite otherwise it is continuous.
  - If there are a limited number of distinct, clearly defined percepts and actions, we say that the environment is discrete.
    - Chess, since there are a fixed number of possible moves on each turn.
    - Taxi driving is continuous.

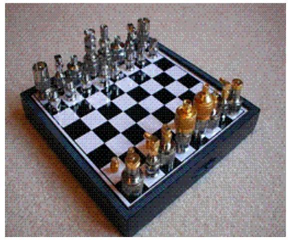
Types of Environment

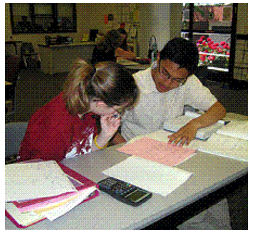
- **Single Agent** vs. **Multi Agent**
  - Is the agent the only thing acting in the world?
  - In the single agent environment there is only one agent
    - A computer software playing crossword puzzle
  - In multiagent systems, there are more than one active agents
    - Video games

vs.

Environment Types Example







	Crossword puzzle	Chess	Taxi driving	Part-picking robot	English tutor
Observable	Yes	Yes	No	No	No
Deterministic	Yes	Yes	No	No	No
Episodic	No	No	No	Yes	No
Static	Yes	Yes	No	No	No
Discrete	Yes	Yes	No	No	Yes
Agents	Single	Multi	Multi	Single	Multi

Structure of Agents

Agent design

- **The environment type largely determines the agent design**
  - ***Fully/partially observable*** => agent requires ***memory*** (internal state)
  - ***Discrete/continuous*** => agent may not be able to enumerate ***all states***
  - ***Stochastic/deterministic*** => agent may have to prepare for ***contingencies***
  - ***Single-agent/multi-agent*** => agent may need to behave ***randomly***

Skeleton Agent Program

- Basic framework for an agent program

```
function SKELETON-AGENT(percept) returns action
 static: memory

 memory   := UPDATE-MEMORY(memory, percept)
 action   := CHOOSE-BEST-ACTION(memory)
 memory   := UPDATE-MEMORY(memory, action)

return action
```

Table Agent Program

- agent program based on table lookup

```
function TABLE-DRIVEN-AGENT(percept) returns action
  static: percepts // initially empty sequence*
          table    // indexed by percept sequences
                   // initially fully specified

  append percept to the end of percepts
  action     := LOOKUP(percepts, table)

return action
```

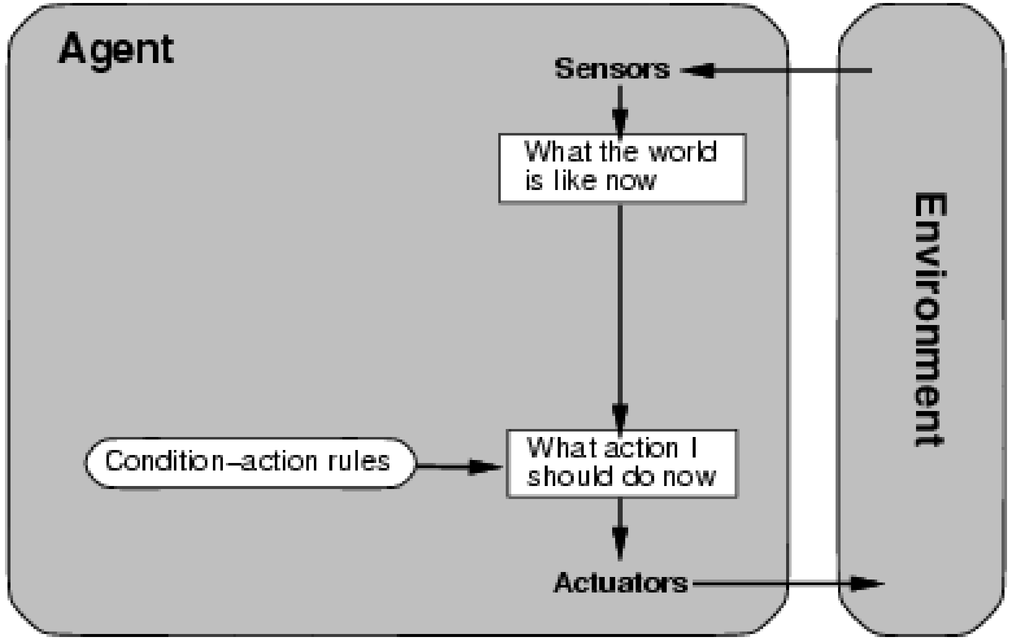
* Note: the storage of percepts requires writeable memory

Structure of Agent

- Different ways of achieving the mapping from percepts to actions
- Types of agents (increasing in generality and ability to handle complex environments)
  - Simple reflex agents
  - Model-based reflex agents
    - keep track of the world
  - Goal-based agents
    - work towards a goal
  - Utility-based agents
  - Learning agent

Simple reflex agents

- Act only on the basis of the **current percept, ignoring the rest of the percept history**.
- The agent function is based on the ***condition-action rule***: if condition then action.
- the vacuum agent is a simple reflex agent, because its decision is based only on the current location and on whether that location contains dirt.
- Fully-observable



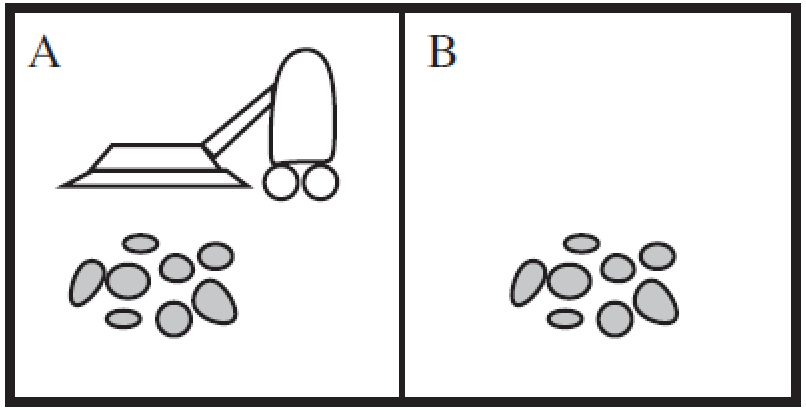
Simple Reflex Agents

- 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 works on Condition-action rule, which means it maps the current state to action.
- Problems for the simple reflex agent design approach:
  - They have very limited intelligence
  - Mostly too big to generate and to store.
  - Not adaptive to changes in the environment.

Example: Vacuum Agent

- **Performance**?
  - 1 point for each square cleaned in time T?
  - #clean squares per time step - #moves per time step?
- **Environment**: vacuum, dirt, multiple areas defined by square regions
- **Actions**: left, right, suck, idle
- **Sensors**: location and contents
  - [A, dirty]

- Rational is not omniscient
  - Environment may be partially observable
- Rational is not clairvoyant
  - Environment may be stochastic
- Thus Rational is not always successful



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

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*

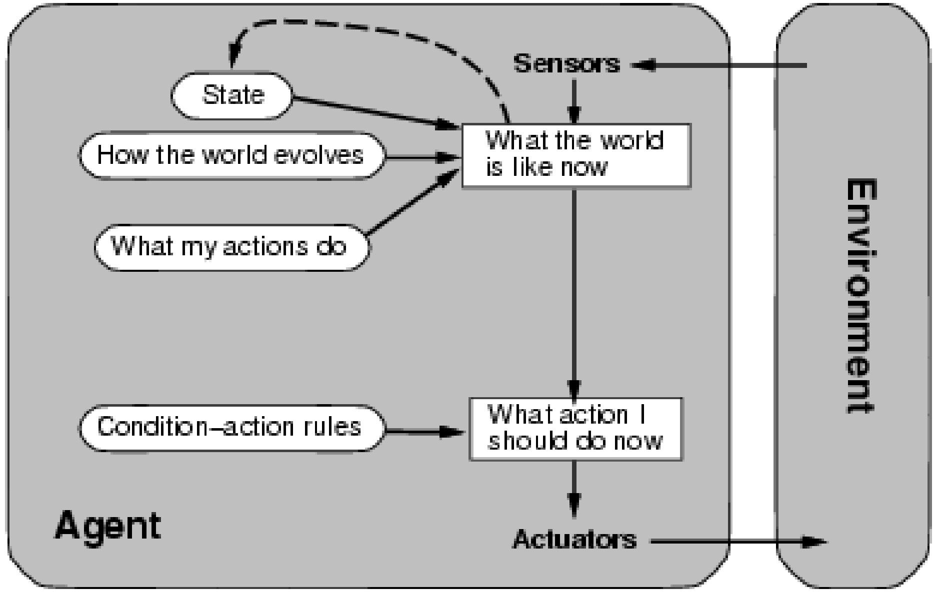
Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

```python
def simple_reflex_agent(percept):
        state = get_state_from_percept(percept)
        rule = match_rule(state, rules)
        action = rule.Action
        return action
```

simple_reflex_agent.py hosted with ❤ by GitHub view raw

Model-based reflex agents

- Keep track of the part of the worls it can't see now.
- Its current state is stored inside the agent ---- **Internal state**
- Percept history and impact of action on the environment can be determined by using internal model.
- It then chooses an action in the same way as reflex agent.
- Partially-observable



Model-Based Reflex Agents

- 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
  - How the agent's action affects the world.

Model-Based 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*

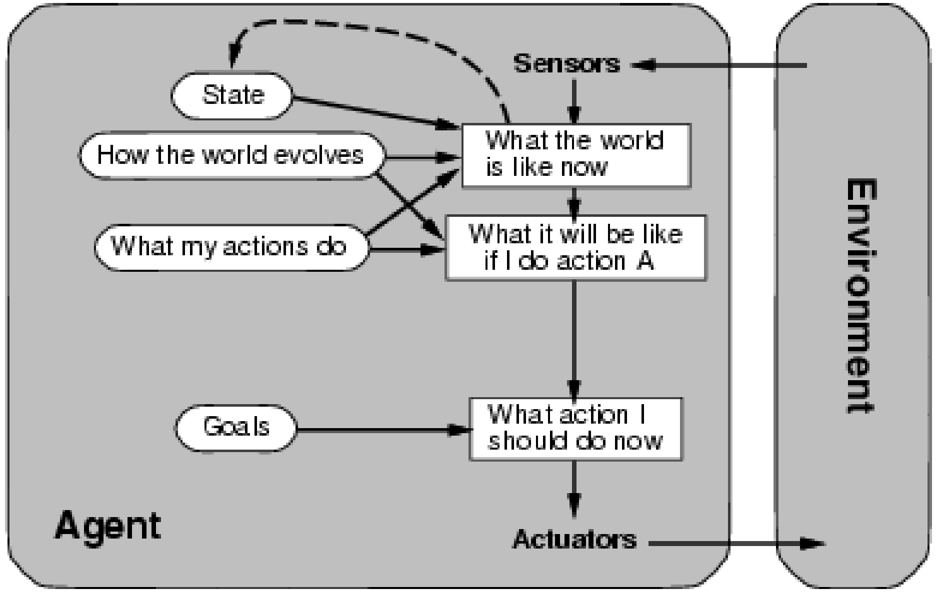
Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

```
def model_based_reflex_agent(percept):
        state = update_state(state, action, percept, model)
        rule = match_rule(state, rules)
        action = rule.Action
        return action
```

model_based_reflex_egent.py hosted with ❤ by GitHub   view raw

Goal-based agents

- Expand on the capabilities of the model-based agents, by using "goal" information.
- Goal information describes situations that are desirable.
- This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state.
- Search and planning



Goal Based Agents

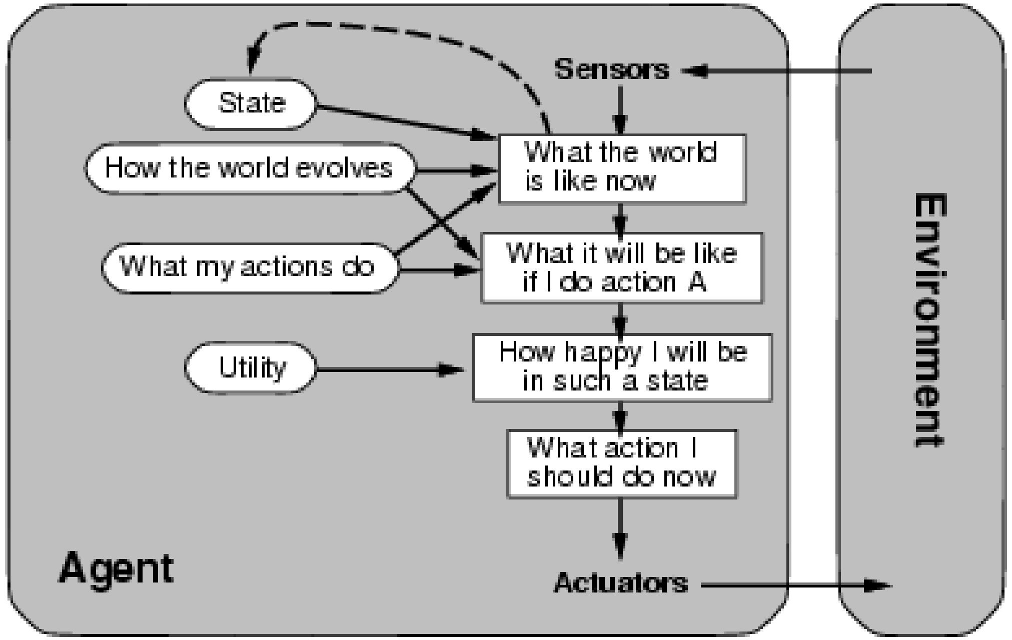
- The knowledge of the current state environment is not always sufficient to decide for an agent to what to do.
- The agent needs to know its goal which describes desirable situations.
- Goal-based agents expand the capabilities of the model-based agent by having the "goal" information.
- They choose 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.

```
function Goal-Based-Agent(percept, goal) returns action
  static : rules, a set of condition-action rules
           state, a description of the current state
  state := Update-State (state, percept)
  rule := Plan-Best-Move(state, rules, goal)
  action := Rule-Action[rule]
  state := Update-State (state, action)
return action
```

Utility-based agents

- We say that if one world state is preferred to another, then it has higher *utility* for the agent.
- Utility is a function that maps a state onto a real number.
  - state → R
- Any rational agent possesses a utility function.

- A utility function maps each state after each action to a real number representing how efficiently each action achieves the goal.

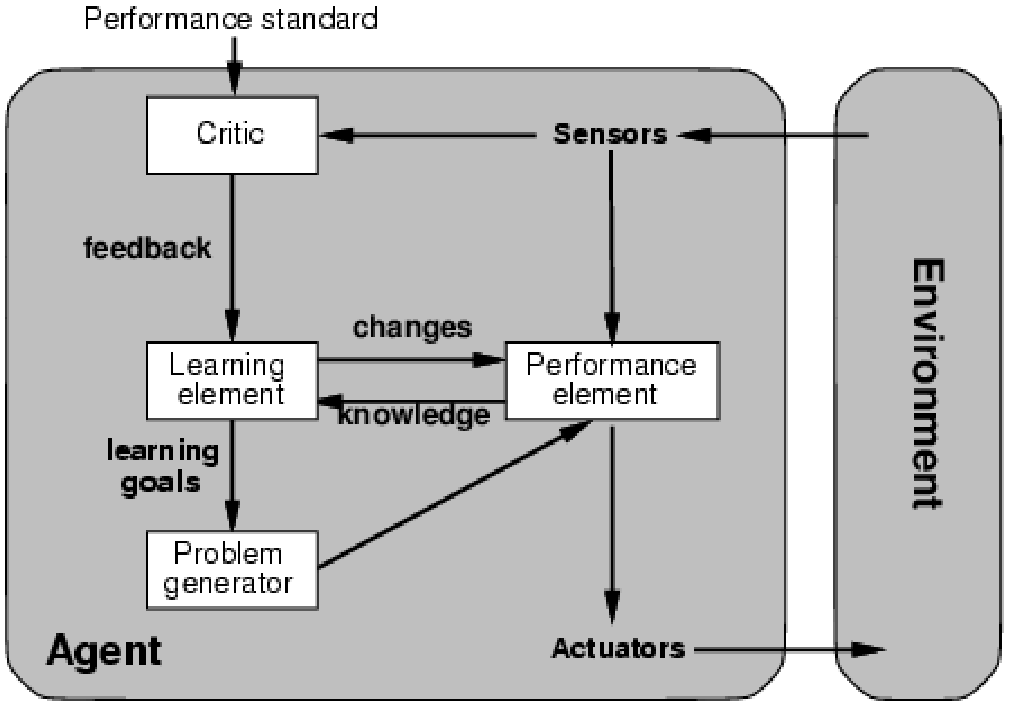


Utility Based Agents

- 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.

Learning Agents

- It allows agents to operate in unknown environments and to become more competent.
- learning element
  - responsible for making improvements
- performance element
  - responsible for selecting actions.
- Uses feedback from the **critic** on how the agent is doing.
- **problem generator**
  - responsible for suggesting actions that will lead to new experiences suggest (exploratory actions).



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.

Intelligent Agents

- **Simple reflex agents** respond directly to percepts
- **model-based reflex agents** maintain internal state to track aspects of the world that are not evident in the current percept.
- **Goal-based agents** act to achieve their goals,
- **utility-based agents** try to maximize their own expected "happiness."