Artificial Intelligence

Lecture: Agent Classifications

Dr. Partha Pakray

Book

Artificial Intelligence: A Modern Approach

Stuart Russell
Peter Norvig

PEARSON Publication

1. Define an agent?
2. What is a rational agent ?
3. What is bounded rationality ?
4. What is an autonomous agent ?
5. Describe the salient features of an agent.

"Surely computers cannot be intelligent; they can do only what their programmers tell them". Is the latter statement true, and does it imply the former?

"Surely computers cannot be intelligent; they can do only what their programmers tell them". Is the latter statement true, and does it imply the former?

The statement that the computers can do only what their programmers tell them is ambiguous. It is true that computers cannot be intelligent because computers are machines that do not have knowledge. Thus, they are not intelligent. They act according to the instructions given by the humans (programmers). They cannot act on their own and they cannot make the decisions by themselves. They are dependent on the algorithms. As they completely depend on the knowledge and instructions of programmers, computers cannot be intelligent.

Understand Types of Environments in Artificial Intelligence

Fully Observable vs Partially-Observable

Real-life Example?

Understand Types of Environments in Artificial Intelligence

Fully Observable vs Partially-Observable

In a fully observable environment, The Agent is familiar with the complete state of the environment at a given time. There will be no portion of the environment that is hidden for the agent.

Real-life Example: While running a car on the road (Environment), The driver (Agent) is able to see road conditions, signboard and pedestrians on the road at a given time and drive accordingly. So Road is a fully observable environment for a driver while driving the car.

in a partially observable environment, The agent is not familiar with the complete environment at a given time.

Real-life Example: Playing card games is a perfect example of a partially-observable environment where a player is not aware of the card in the opponent's hand. Why partially-observable? Because the other parts of the environment, e.g opponent, game name, etc are known for the player (Agent).

Deterministic vs Stochastic

Real-life Example?

Deterministic vs Stochastic

Deterministic are the environments where the next state is observable at a given time. So there is no uncertainty in the environment.

Real-life Example: The traffic signal is a deterministic environment where the next signal is known for a pedestrian (Agent)

The Stochastic environment is the opposite of a deterministic environment. The next state is totally unpredictable for the agent. So randomness exists in the environment.

Real-life Example: The radio station is a stochastic environment where the listener is not aware about the next song or playing a soccer is stochastic environment.

Episodic vs Sequential

Real-life Example?

Episodic vs Sequential

Episodic is an environment where each state is independent of each other. The action on a state has nothing to do with the next state.

Real-life Example: A support bot (agent) answer to a question and then answer to another question and so on. So each question-answer is a single episode.

The sequential environment is an environment where the next state is dependent on the current action. So agent current action can change all of the future states of the environment.

Real-life Example: Playing tennis is a perfect example where a player observes the opponent's shot and takes action.

Static vs Dynamic

Real-life Example?

Static vs Dynamic

The Static environment is completely unchanged while an agent is precepting the environment.

Real-life Example: Cleaning a room (Environment) by a dry-cleaner reboot (Agent) is an example of a static environment where the room is static while cleaning.

Dynamic Environment could be changed while an agent is precepting the environment. So agents keep looking at the environment while taking action.

Real-life Example: Playing soccer is a dynamic environment where players' positions keep changing throughout the game. So a player hit the ball by observing the opposite team.

Discrete vs Continuous

Real-life Example?

Discrete vs Continuous

Discrete Environment consists of a finite number of states and agents have a finite number of actions.

Real-life Example: Choices of a move (action) in a tic-tac game are finite on a finite number of boxes on the board (Environment).

While in a Continuous environment, the environment can have an infinite number of states. So the possibilities of taking an action are also infinite.

Real-life Example: In a basketball game, the position of players (Environment) keeps changing continuously and hitting (Action) the ball towards the basket can have different angles and speed so infinite possibilities.

Single Agent vs Multi-Agent

Real-life Example?

Single Agent vs Multi-Agent

Single agent environment where an environment is explored by a single agent. All actions are performed by a single agent in the environment.

Real-life Example: Playing tennis against the ball is a single agent environment where there is only one player.

If two or more agents are taking actions in the environment, it is known as a multi-agent environment.

Real-life Example: Playing a soccer match is a multi-agent environment.

Characterizing a Task Environment

- Must first specify the setting for intelligent agent design.
- **PEAS: Performance measure, Environment, Actuators, Sensors**
- **Example:** the task of designing a self-driving car

- **Performance measure** **Safe, fast, legal, comfortable trip**
- **Environment** **Roads, other traffic, pedestrians**
- **Actuators** **Steering wheel, accelerator, brake, signal, horn**
- **Sensors** **Cameras, LIDAR (light/radar), speedometer, GPS, odometer engine sensors, keyboard**

Assignment-I
(through google classroom)

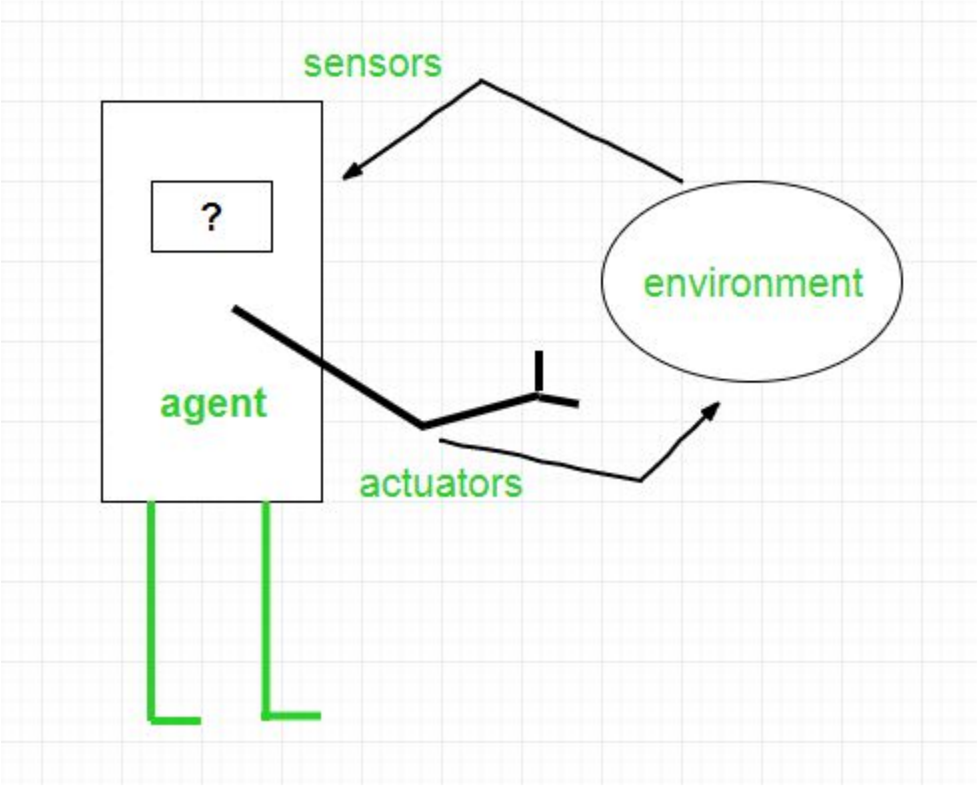
Assignment 1: ***For Flipkart shopping agent, what is the Agent Type, Environment, Actuators, Performance Measure, Sensors, Agent Architecture?***

Assignment 2: ***Suppose you want to develop an automated taxi, so what will be the Agent Type, Environment, Actuators, Performance Measure, Sensors?***

Assignment 3: Write the names in side the table

Task Environment	Observable	Deterministic	Episodic	Static	discrete	Agents
Crossword puzzle						
Chess with a clock						
Poker						
Taxi driving						
Medical Diagnosis						
Image Analysis						
Interactive Tutoring system (English Tutor)						

AI Agent



Structure of Agents

Agent = architecture + program

Architecture = some sort of computing device (sensors + actuators)

(Agent) Program = some function that implements the agent mapping = “?”

Agent Program = Job of AI

Agent architectures

- *Table based agent*
- *Percept based agent or reflex agent*
- *State-based Agent or model-based reflex agent*
- *Goal-based Agent*
- *Utility-based Agent*
- *Learning Agent*

Table based agent

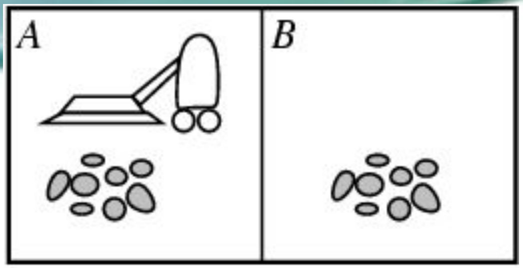
- In table based agent the action is looked up from a table based on information about the agent's percepts.
- A table is simple way to specify a mapping from percepts to actions. The mapping is implicitly defined by a program.
- The mapping may be implemented by a rule based system, by a neural network or by a procedure.

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

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

Disadvantages:

- The tables may become very large.
- Learning a table may take a very long time, especially if the table is large.

Toy example:
Vacuum world.



Percepts: robot senses it's location and "cleanliness."

So, location and contents, e.g., [A, Dirty], [B, Clean].

With 2 locations, we get **4 different possible sensor inputs.**

Actions: *Left*, *Right*, *Suck*, *NoOp*

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*
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Percept based agent or reflex agent

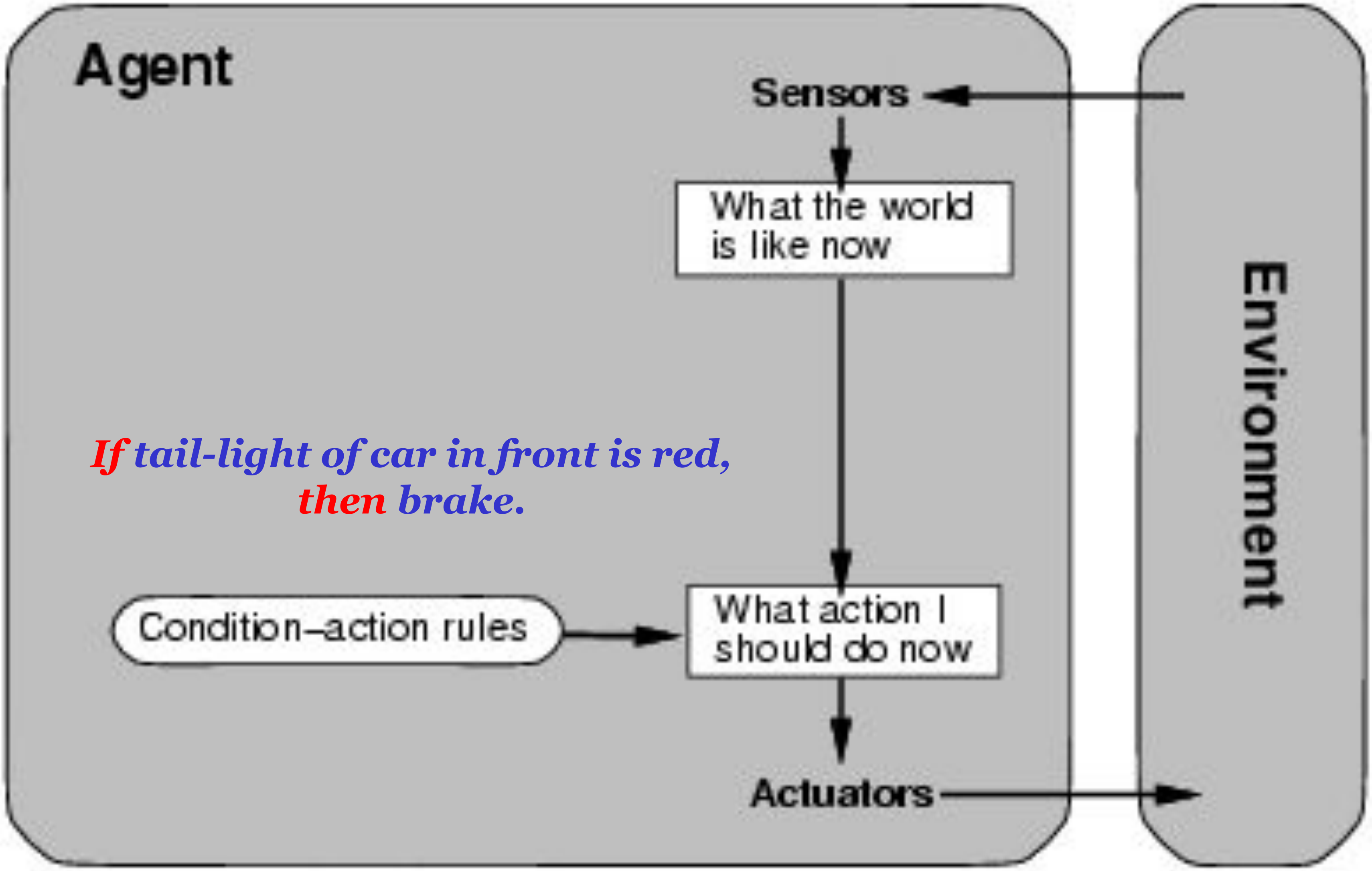
- In percept based agents,
  - information comes from **sensors - percepts**
  - changes the agents current **state of the world**
  - triggers **actions** through the **effectors**

- Such agents are called **reactive agents** or **stimulus-response agents**.

  Reactive agents have no notion of history. The current state is as the sensors see it right now. The action is based on the current percept only.

```
function SIMPLE-REFLEX-AGENT(percept) returns action
   static: rules, a set of condition-action rules

   state ← INTERPRET-INPUT(percept)
   rule ← RULE-MATCH(state, rules)
   action ← RULE-ACTION[rule]
   return action
```

Agent selects actions on the basis of *current* percept only.



A Simple Reflex Agent in Nature

percepts
(size, motion)

RULES:

(1) If small moving object,
then activate SNAP

(2) If large moving object,
then activate AVOID and inhibit SNAP

ELSE (not moving) then NOOP

needed for completeness

Action: SNAP or AVOID or NOOP

- The following are some of the characteristics of percept-based agents.
  - Efficient
  - No internal representation for reasoning, inference.
  - No strategic planning, learning.
  - Percept-based agents are not good for multiple, opposing, goals.

State-based Agent or model-based reflex agent

- state based agent works as follows:
  - information comes from **sensors – percepts**
  - based on this, the agent changes the current state of the world
  - based on state of the world and knowledge (memory), it triggers actions through the effectors
  - E.g., driving a car and changing lane

Requiring two types of knowledge

- How the world evolves independently of the agent
- How the agent's actions affect the world

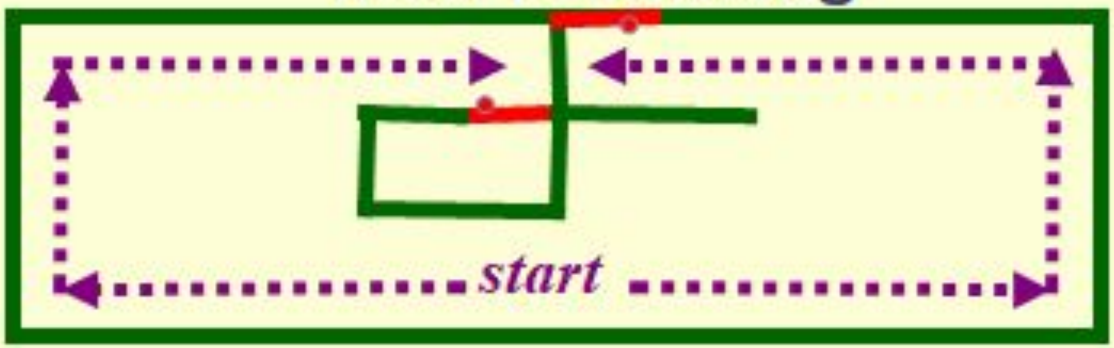
```
function REFLEX-AGENT-WITH-STATE(percept) returns action
   static: state, a description of the current world state
           rules, a set of condition-action rules

   state ← UPDATE-STATE(state, percept)
   rule ← RULE-MATCH(state, rules)
   action ← RULE-ACTION[rule]
   state ← UPDATE-STATE(state, action)
   return action
```

The agent is with memory

Example Table Agent With Internal State

IF	THEN
Saw an object ahead, and turned right, and it's now clear ahead	Go straight
Saw an object Ahead, turned right, and object ahead again	Halt
See no objects ahead	Go straight
See an object ahead	Turn randomly

Example Reflex Agent With Internal State: **Wall-Following**



Actions: left, right, straight, open-door

Rules:

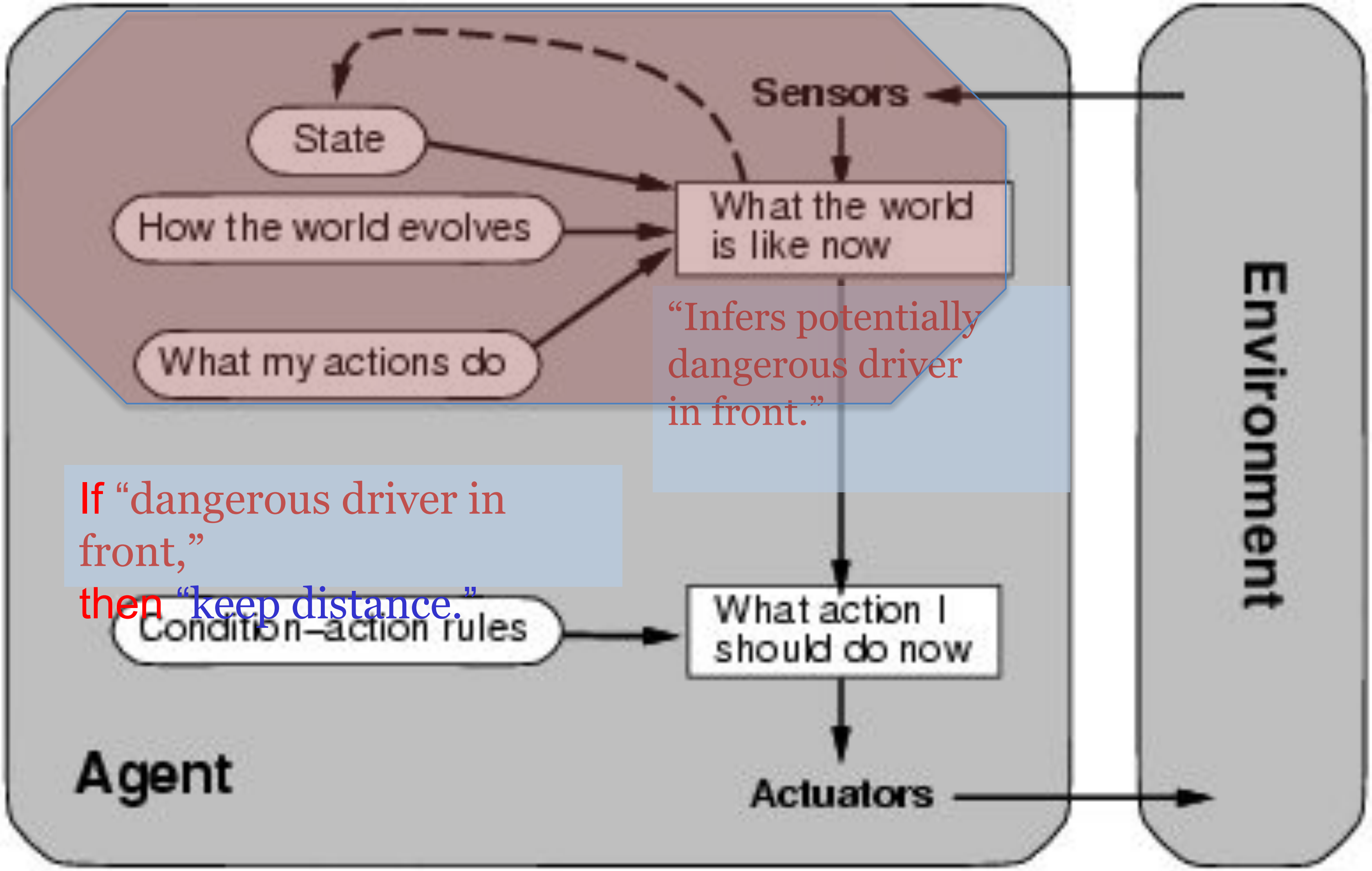
1. If open(left) & open(right) and open(straight) then choose randomly between right and left
2. If wall(left) and open(right) and open(straight) then straight
3. If wall(right) and open(left) and open(straight) then straight
4. If wall(right) and open(left) and wall(straight) then left
5. If wall(left) and open(right) and wall(straight) then right
6. If wall(left) and door(right) and wall(straight) then open-door
7. If wall(right) and wall(left) and open(straight) then straight.
8. (Default) Move randomly

Model-based reflex agents

[Here] **Module: Logical Agents**

Representation and Reasoning: Part III/IV R&N

How detailed?

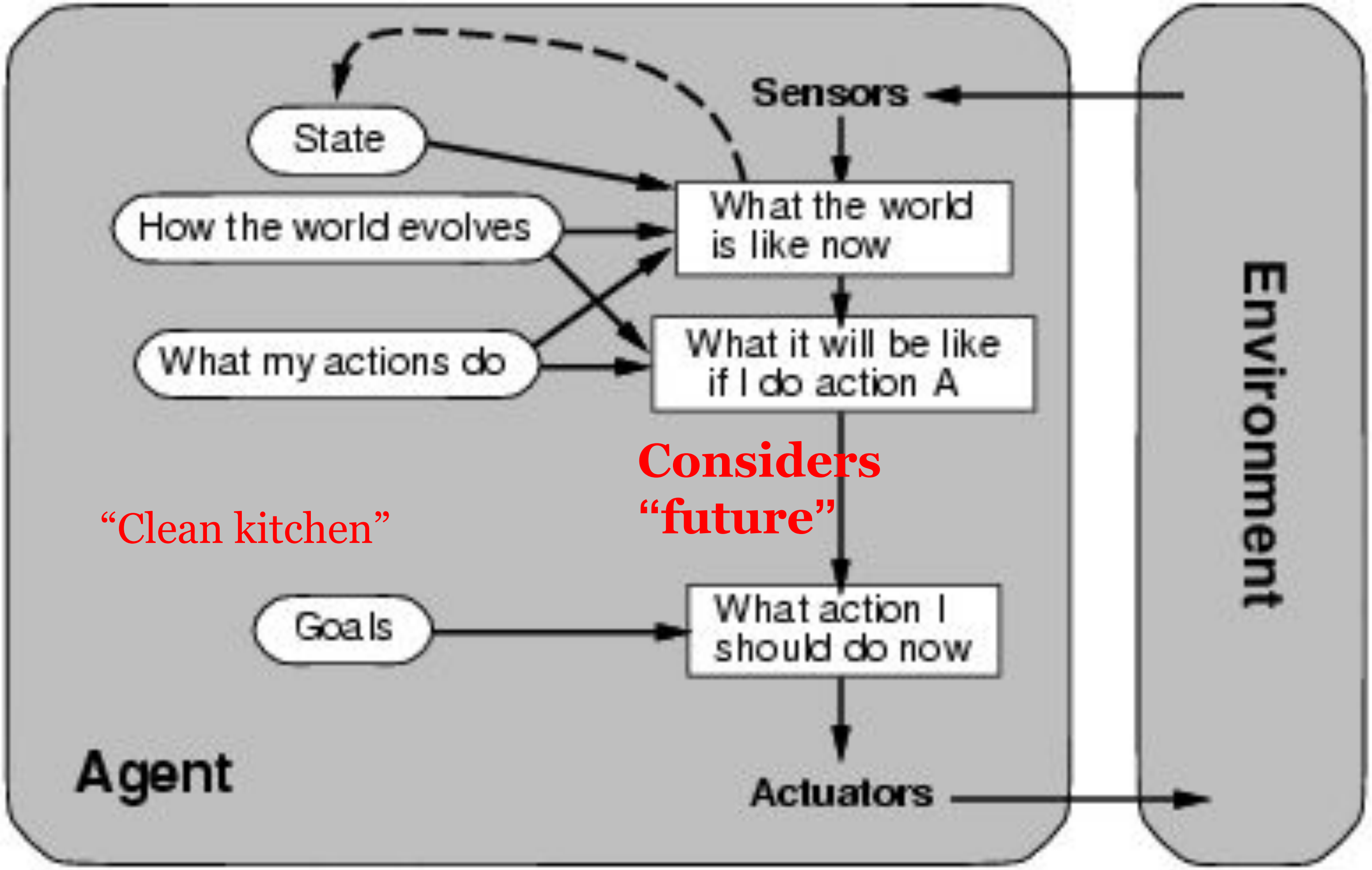


Goal-based Agent

- Agent Actions will depend upon its goal.
- Such agents work as follows:
    - information comes from **sensors – percepts**
    - changes the agents current state of the world
    - based on **state of the world** and **knowledge (memory)** and **goals/intentions**, it chooses **actions** and does them through the **effectors**.
- Goal formulation based on the current situation is a way of solving many problems and search is a universal problem solving mechanism in AI.
- The sequence of steps required to solve a problem is not known **a priori** and must be determined by **a systematic exploration** of the alternatives.

Goal-based agents

Module: Problem Solving



Agent keeps track of the world state as well as set of goals it's trying to achieve: chooses actions that will (eventually) lead to the goal(s).

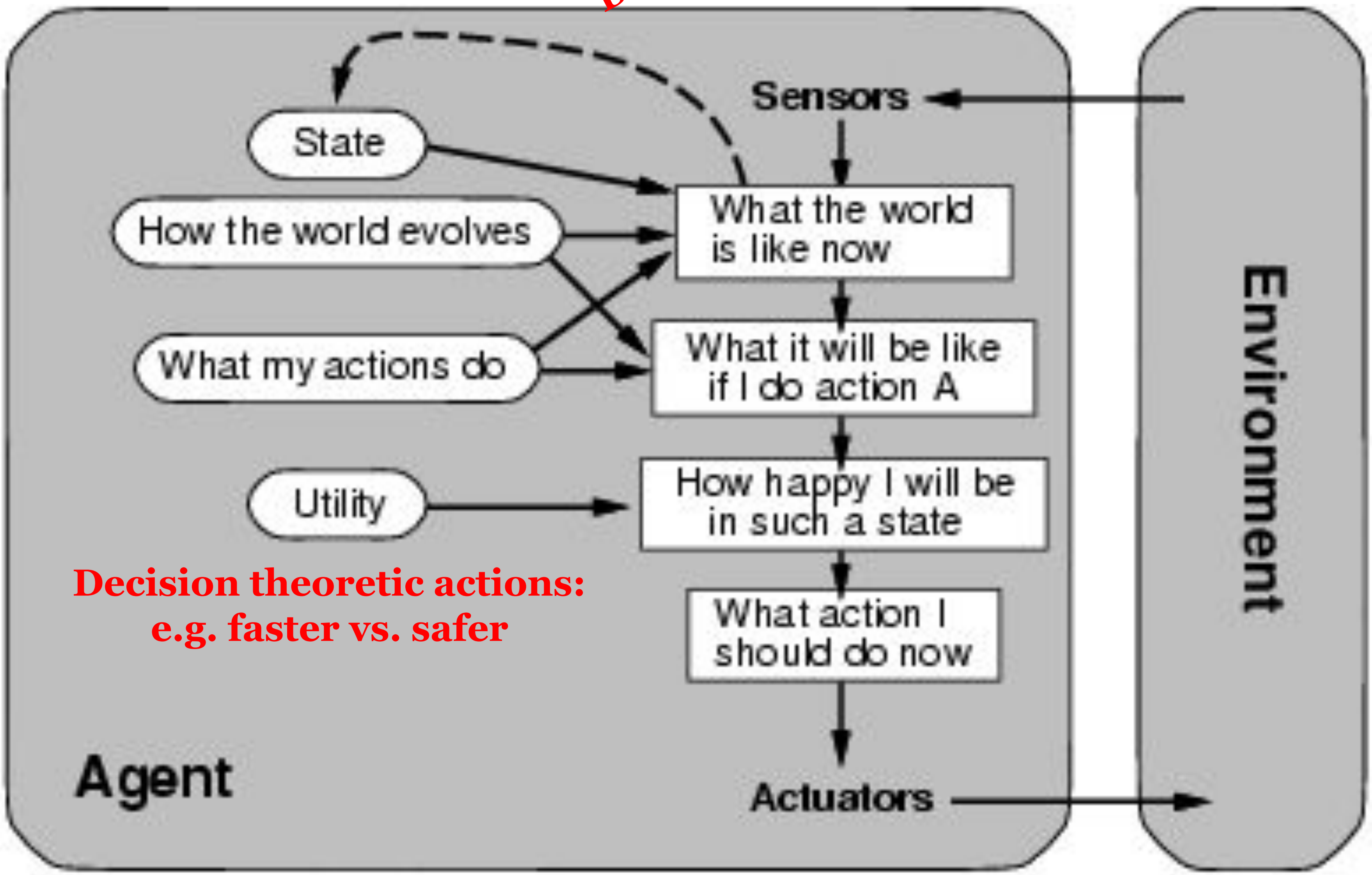
More flexible than reflex agents □ may involve search and planning

Utility-based Agent

- Utility based agents provides a more general agent framework.
- different preferences for the different goals
- A utility function maps a state or a sequence of states to a real valued utility.
- The agent acts so as to maximize expected utility
- look for a **quicker, safer, cheaper trip** to reach a destination. Agent happiness should be taken into consideration.
- **Utility describes how "happy" the agent is.**

Utility-based agents

Module:
Decision Making



Decision theoretic actions:
e.g. faster vs. safer

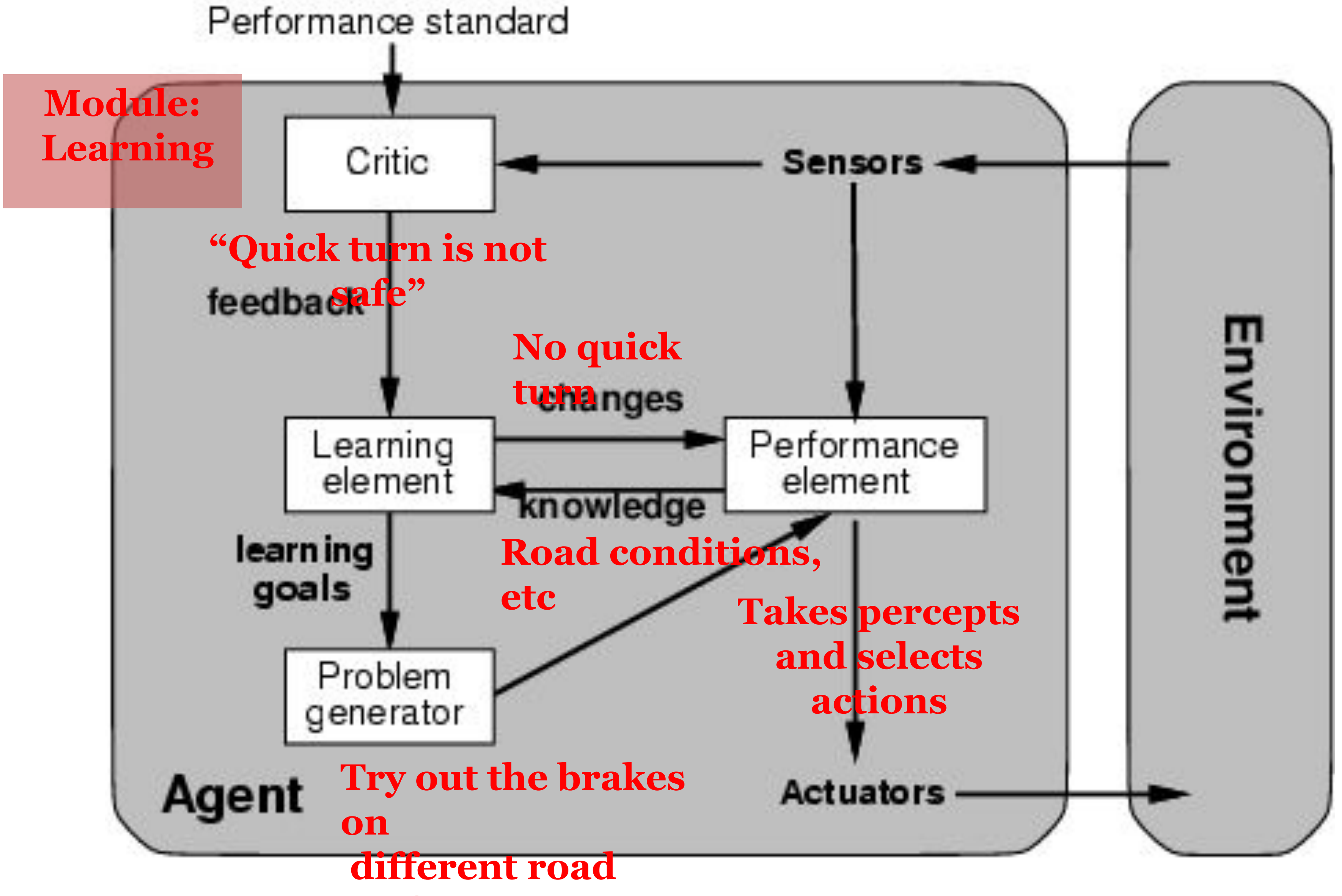
Learning Agent

- Learning allows an agent to operate in initially unknown environments.
- The learning element modifies the performance element.
- Learning is required for true autonomy

More complicated when agent needs to learn utility information:
Reinforcement learning
(based on action payoff)

Learning agents

Adapt and improve over time



Conclusion

- An **agent** perceives and acts in an environment, has an architecture, and is implemented by an agent program.
- An **ideal agent** always chooses the action which maximizes its expected performance, given its percept sequence so far.
- An **autonomous agent** uses its own experience rather than built-in knowledge of the environment by the designer.
- An agent program maps from percept to action and updates its internal state.
  - Reflex agents respond immediately to percepts.
  - Goal-based agents act in order to achieve their goal(s).
  - Utility-based agents maximize their own utility function.
- Representing knowledge is important for successful agent design.
- The most challenging environments are partially observable, stochastic, sequential, dynamic, and continuous, and contain multiple intelligent agents.

Thank You!

Any Questions?