THE STRUCTURE OF AGENTS

2.4.1 Agent programs

- **agent program-**which takes the current percept as input
- **agent function-**which takes the entire percept history

```
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.

Drawbacks:

- **Table lookup** of percept-action pairs defining all possible condition-action rules necessary to interact in an environment
- **Problems**
  - Too big to generate and to store (Chess has about 10^120 states, for example)
  - No knowledge of non-perceptual parts of the current state
  - Not adaptive to changes in the environment; requires entire table to be updated if changes occur
  - Looping: Can't make actions conditional
- Take a long time to build the table
- No autonomy
- Even with learning, need a long time to learn the table entries

Some Agent Types

- **Table-driven agents**
  - use a percept sequence/action table in memory to find the next action. They are implemented by a (large) **lookup table**.
- **Simple reflex agents**
  - are based on **condition-action rules**, implemented with an appropriate production system. They are stateless devices which do not have memory of past world states.
- **Agents with memory**
  - have **internal state**, which is used to keep track of past states of the world.
- **Agents with goals**
  - are agents that, in addition to state information, have **goal information** that describes desirable situations. Agents of this kind take future events into consideration.
- **Utility-based agents**
  - base their decisions on **classic axiomatic utility theory** in order to act rationally.

Simple Reflex Agent

- These agents select actions on the basis of the *current* percept, ignoring the rest of the percept history
- For example, the vacuum agent -its decision is based only on the current location and on whether that contains dirt.

```
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
```

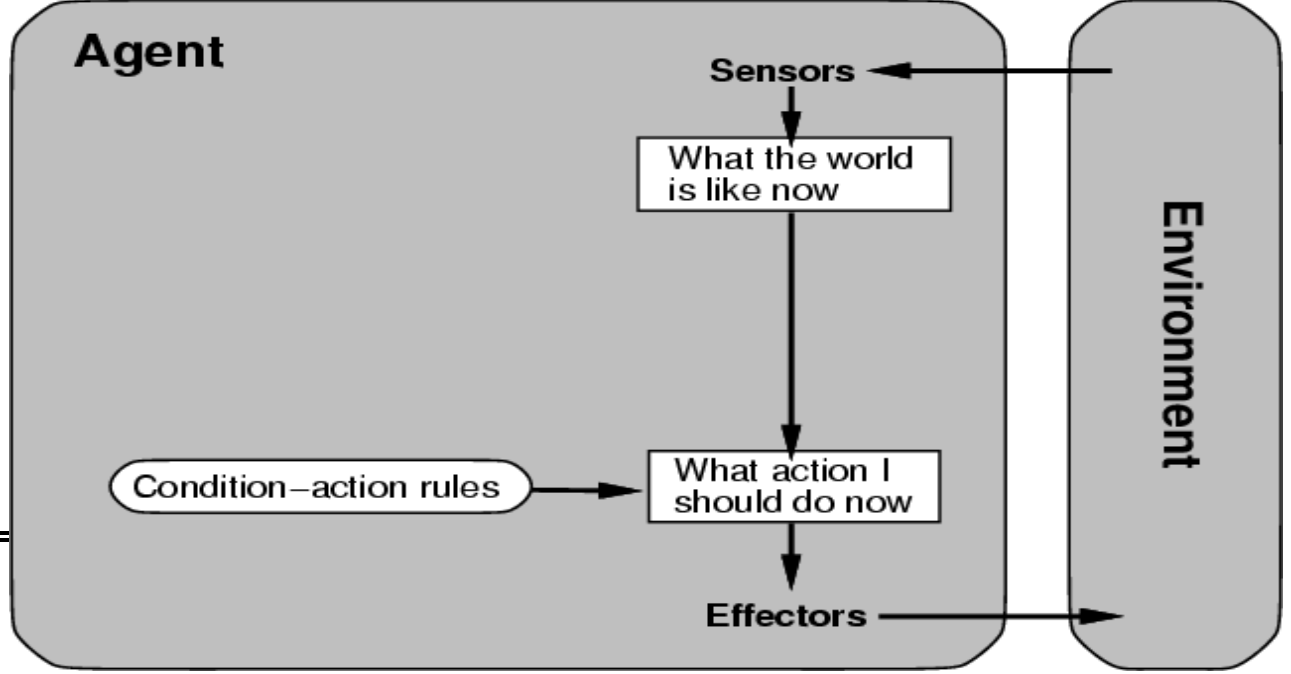


Figure 2.10 A simple reflex agent. It acts according to a rule the current state, as defined by the percept.

Characteristics

Only works if the environment is fully observable.

Lacking history, easily get stuck in infinite loops

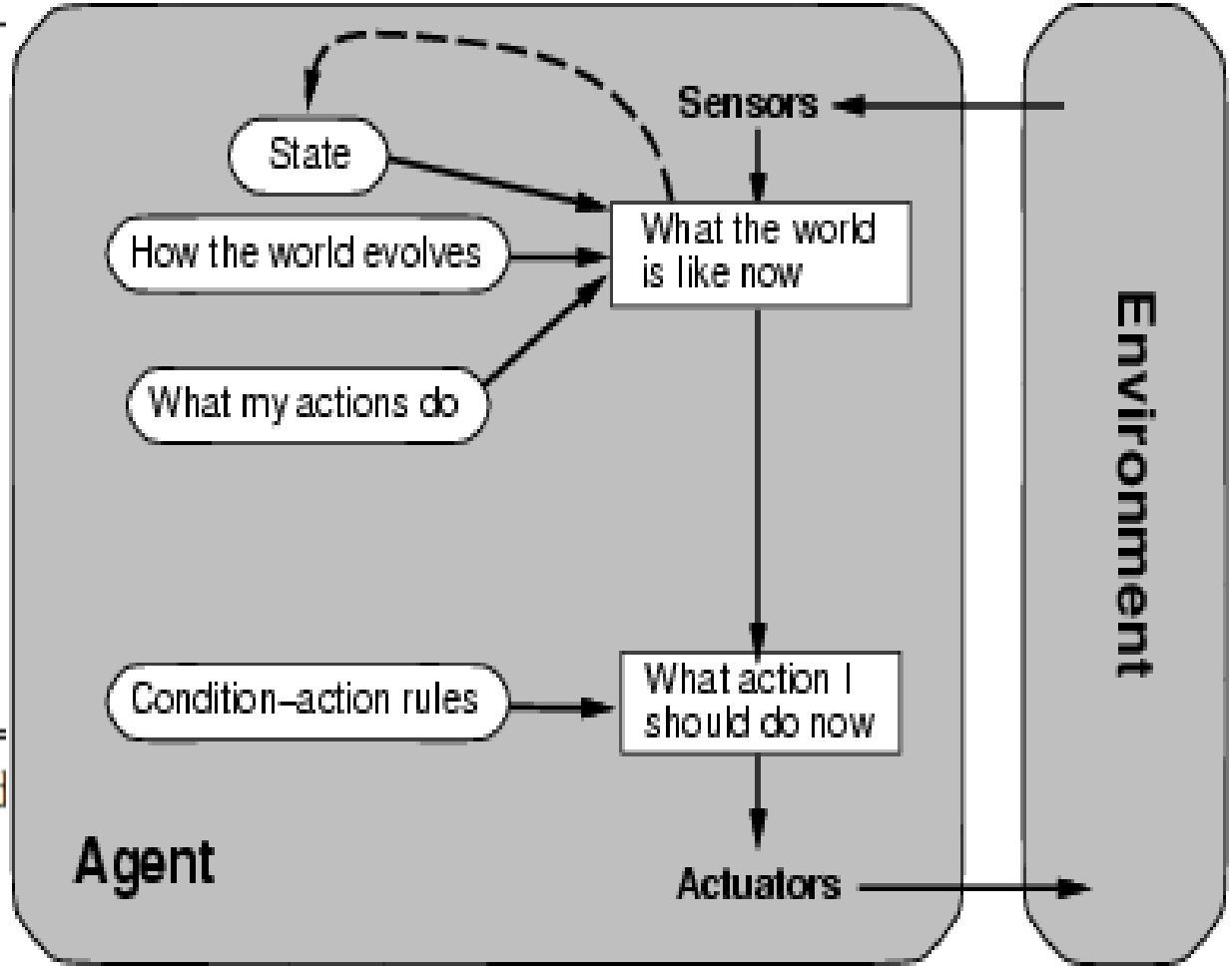
One solution is to randomize actions

Model-based reflex agents

- The most effective way to handle partial observability is for the agent to *keep track of the part of the world it can't see now.*
- Sol: maintain some sort of **internal state** that depends on the percept history -reflects - unobserved aspects of the current state.
- **Two type of knowledge:**
  1. how the world evolves independently of the agent- overtaking car
  2. how the agent's own actions affect the world-agent turns the steering wheel clockwise

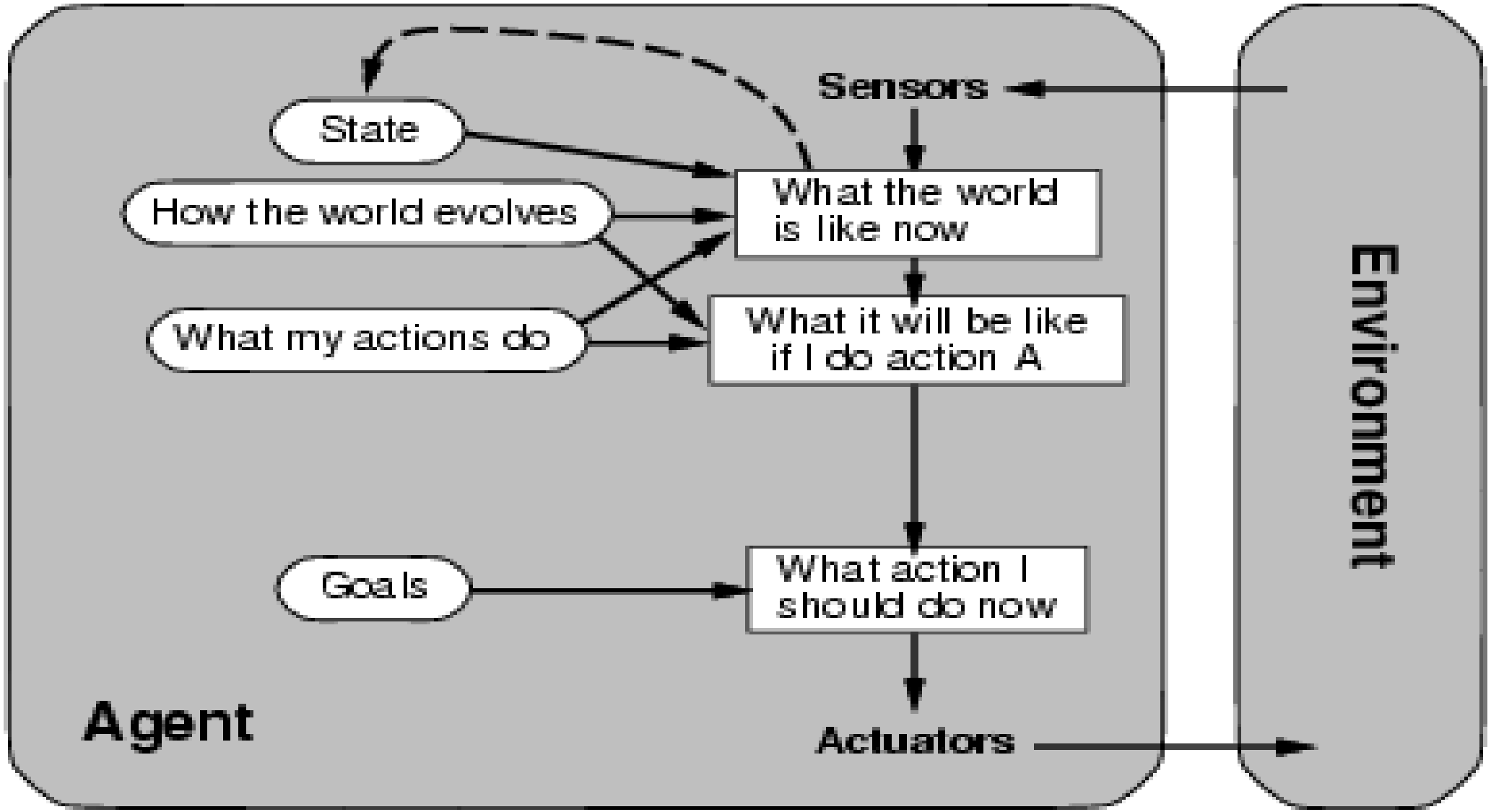
```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, 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.



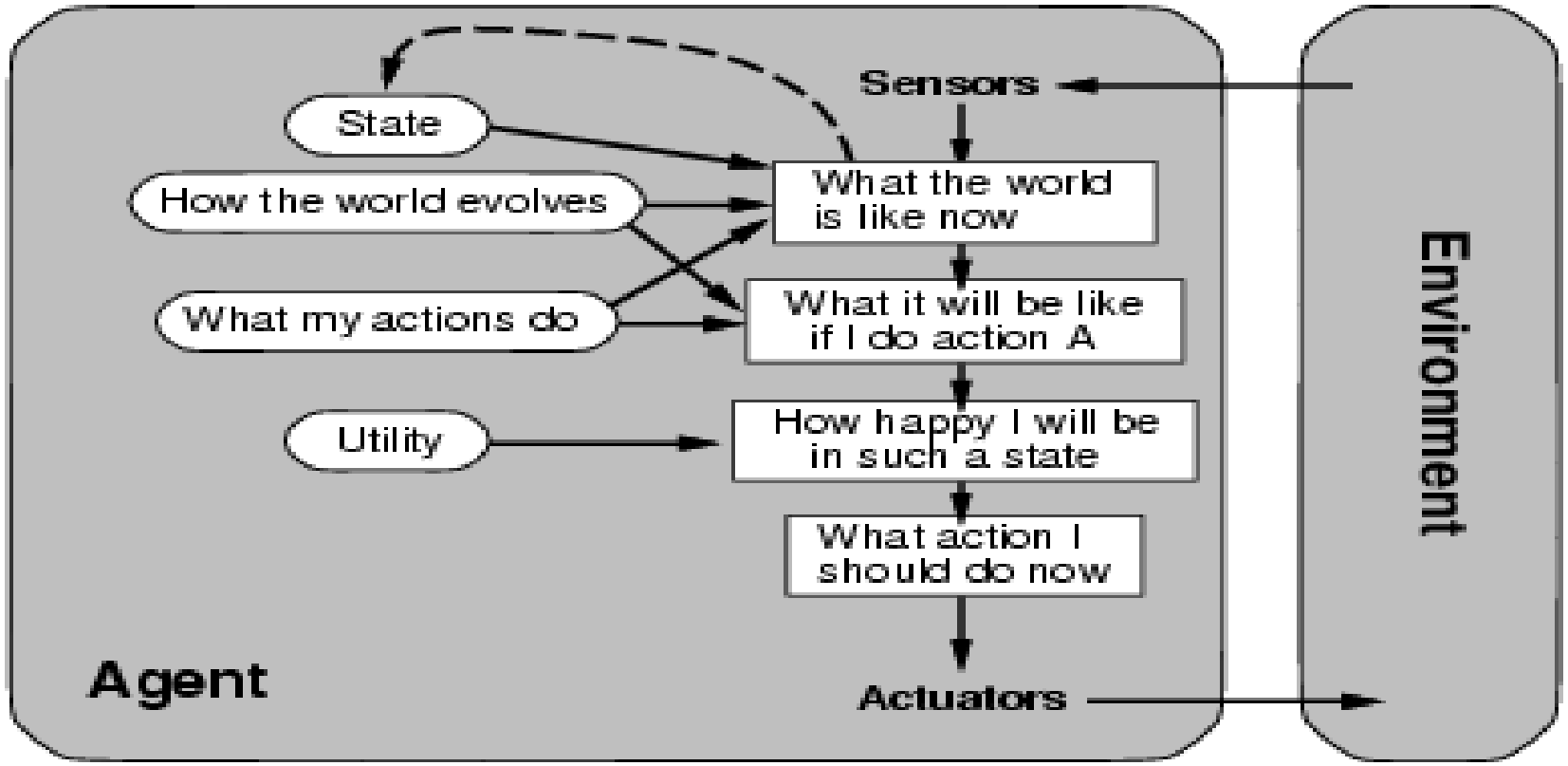
Goal-based agents

- Knowing about the current state of the environment is not always enough to decide what to do.
- For example- at a road junction ?
- In other words, as well as a current state description, the agent needs some sort of **goal** information that describes situations that are desirable-for example, being at the passenger's destination.
- Some times it will be tricky with long sequence and twist to their goals, then they have to use ‘search’ & ‘planning’. Different from condition-action rule.


Sensors
State
How the world evolves
What the world is like now
What my actions do
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

Utility-based agents

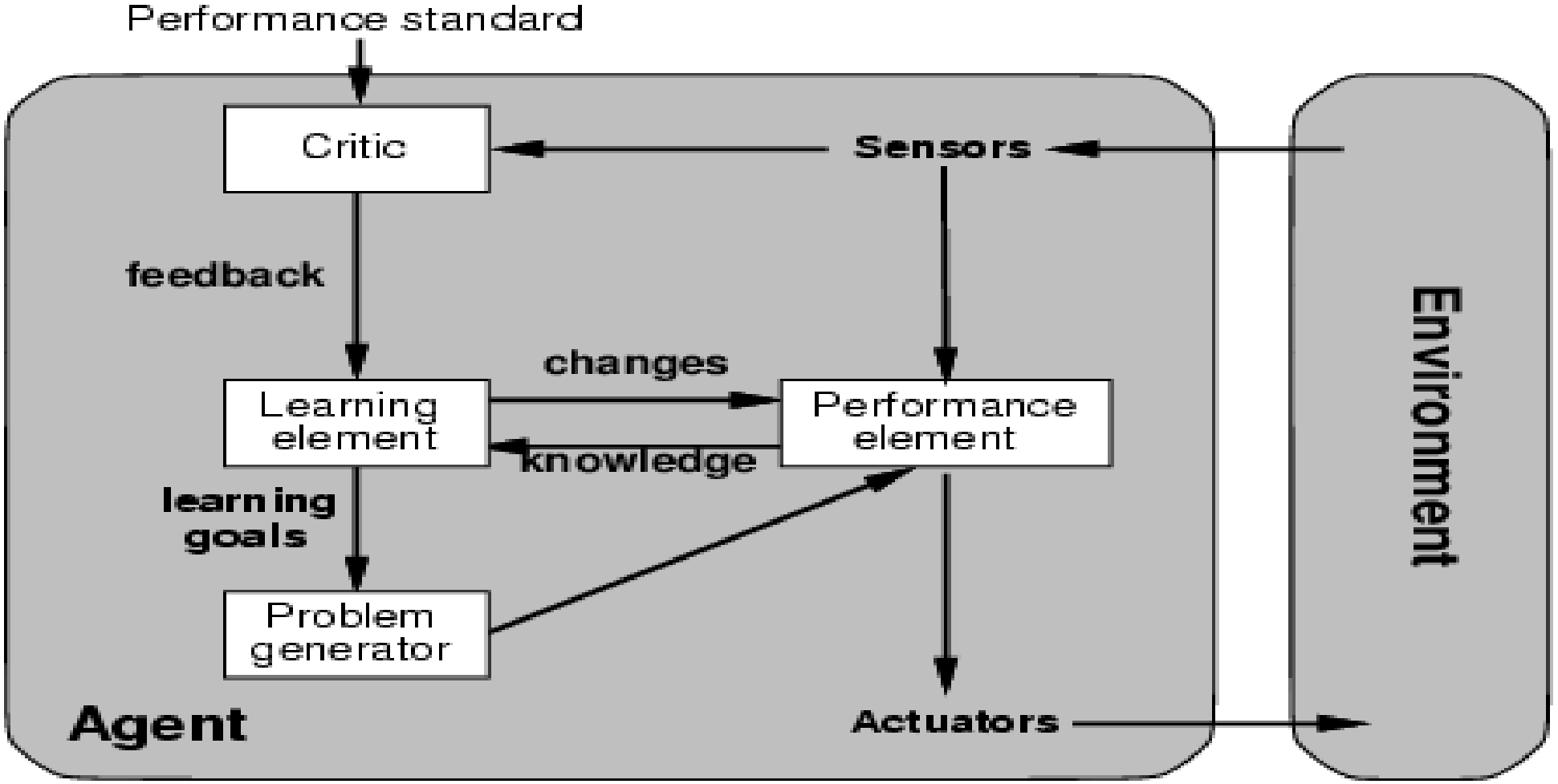
- Goals alone are not really enough to generate high-quality behavior in most environments.
- For example, there are many action sequences -taxi to its destination - achieving the goal-
- Goals just provide a crude binary distinction between "happy" and "unhappy" states
- **performance measure? – preference**



A model-based, utility-based agent. It 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 computed by averaging over all possible outcome states, weighted by the probability of the outcome.

- Certain goals can be reached in different ways.
  - Some are better, have a higher utility.
- If utility fun and performance measure are in agreement- agent chooses action to maximize its utility- rational
- Conflicting goals: (speed & safety)- utility function specifies the trade off-
- Select appropriately between several goals based on likelihood of success.

Learning agents



learning element, which is responsible for making improvements
performance element, which is responsible for selecting external actions
problem generator. It is responsible for suggesting actions that will lead to new and **informative experiences**

Summary: Intelligent Agents

- An **agent** perceives and acts in an environment, has an architecture, and is implemented by an agent program.
- Task environment – **PEAS (P**erformance**, E**nvironment, **A**ctuators, **S**ensors**)**
- The most challenging environments are inaccessible, nondeterministic, dynamic, and continuous.
- An **ideal agent** always chooses the action which maximizes its expected performance, given its percept sequence so far.
- An **agent program** maps from percept to action and updates internal state.
  - **Reflex agents** respond immediately to percepts.
    - simple reflex agents
    - model-based reflex agents
  - **Goal-based agents** act in order to achieve their goal(s).
  - **Utility-based agents** maximize their own utility function.
- All agents can improve their performance through **learning.**