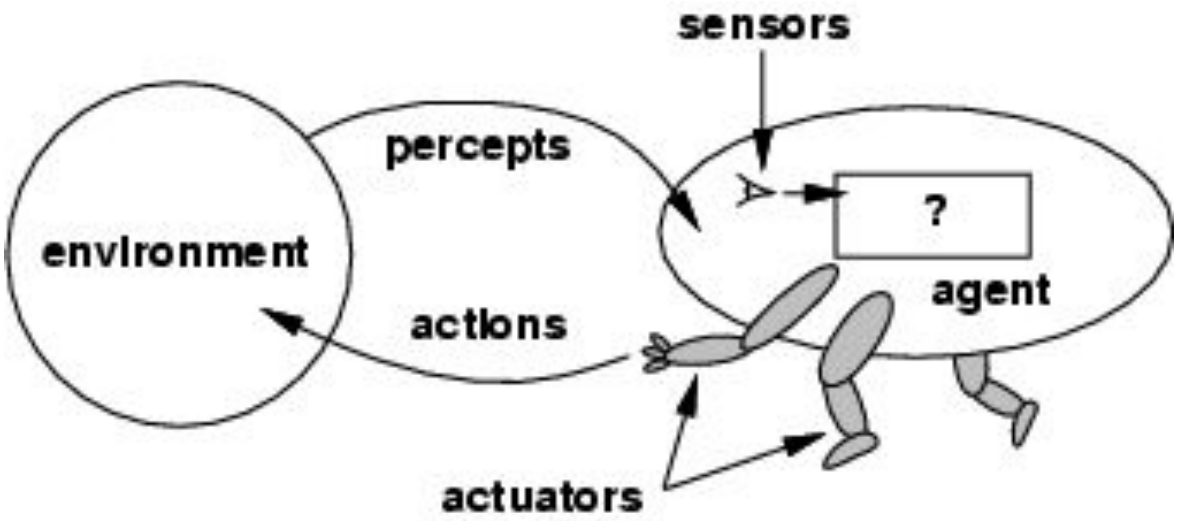
Intelligent Agents

Chapter 2

Agents

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators



Examples

- 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
- Software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on environment by displaying on screen, writing files and sending network packets

Important Terms

percept sequence
Complete History of everything the agent has Perceived.

The agent function maps from percept sequence to actions

Agent Function – Abstract mathematical Description

Describes Agent's Behavior

Agent Program – Implementation of Agent function running on agent Architecture

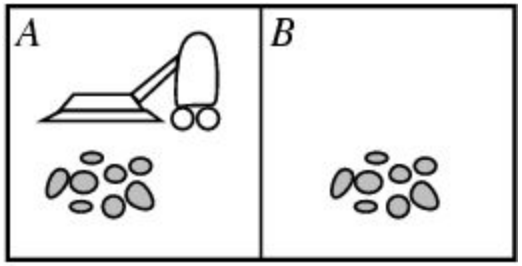
Architecture – Some sort of Computing Device with Physical Sensors and Actuators

agent = architecture + program

Agents and environments

- **Percept sequence** is the complete history of everything the agent has ever perceived.
- *Agents choice of action at any given instant can depend on the entire percept sequence observed to date*
- *Percept sequence and Actions can be tabulated*

Vacuum-cleaner world



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

A vacuum-cleaner agent

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
⋮	⋮

```
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
```

Rational agents

- 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**

- **Performance measure**: An objective criterion to measure success of an agent's behavior
- When agent is brought in an environment, it generates sequence of actions according to the percept it receives
- This sequence of actions cause environment to go through a sequence of states. If sequence is desirable, then agent has performed well.

Rational Agent can Maximize the Performance measure

Rational agents

- Rationality
  - 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
- 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.

Rational agents

- Rationality is distinct from **omniscience**
- Omniscient agent knows the actual outcome of its action and can act accordingly.
- Impossible in reality
- Rationality is not same as perfection. Rationality maximizes expected performance while perfection maximizes actual performance

- Agents can perform actions in order to modify future percepts so as to obtain useful information (**information gathering**, exploration)
- Rational agent not only to gather information but also to **learn** as much as possible from what it percepts

dung beetle

sphex wasp

- An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt)

The nature of task environments

- PEAS: Performance measure, Environment, Actuators, Sensors
- Must first specify the setting for intelligent agent design
- Consider, e.g., the task of designing an automated taxi driver:
  - Performance measure
  - Environment
  - Actuators
  - Sensors

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

PEAS

- Agent: Medical diagnosis system
- Performance measure:
  - Healthy patient, minimize costs
- Environment:
  - Patient, hospital, staff
- Actuators:
  - Screen display (questions, tests, diagnoses, treatments, referrals)
- Sensors:
  - Keyboard (entry of symptoms, findings, patient's answers)

PEAS

- 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

Environment types

- Fully observable (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.
  - Agent need not maintain any internal state
- Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)
- Episodic (vs. sequential): 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.

Environment types

- Static (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is semidynamic if the environment itself does not change with the passage of time but the agent's performance score does)
- Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions.
- Single agent (vs. multiagent): An agent operating by itself in an environment.

Environment types

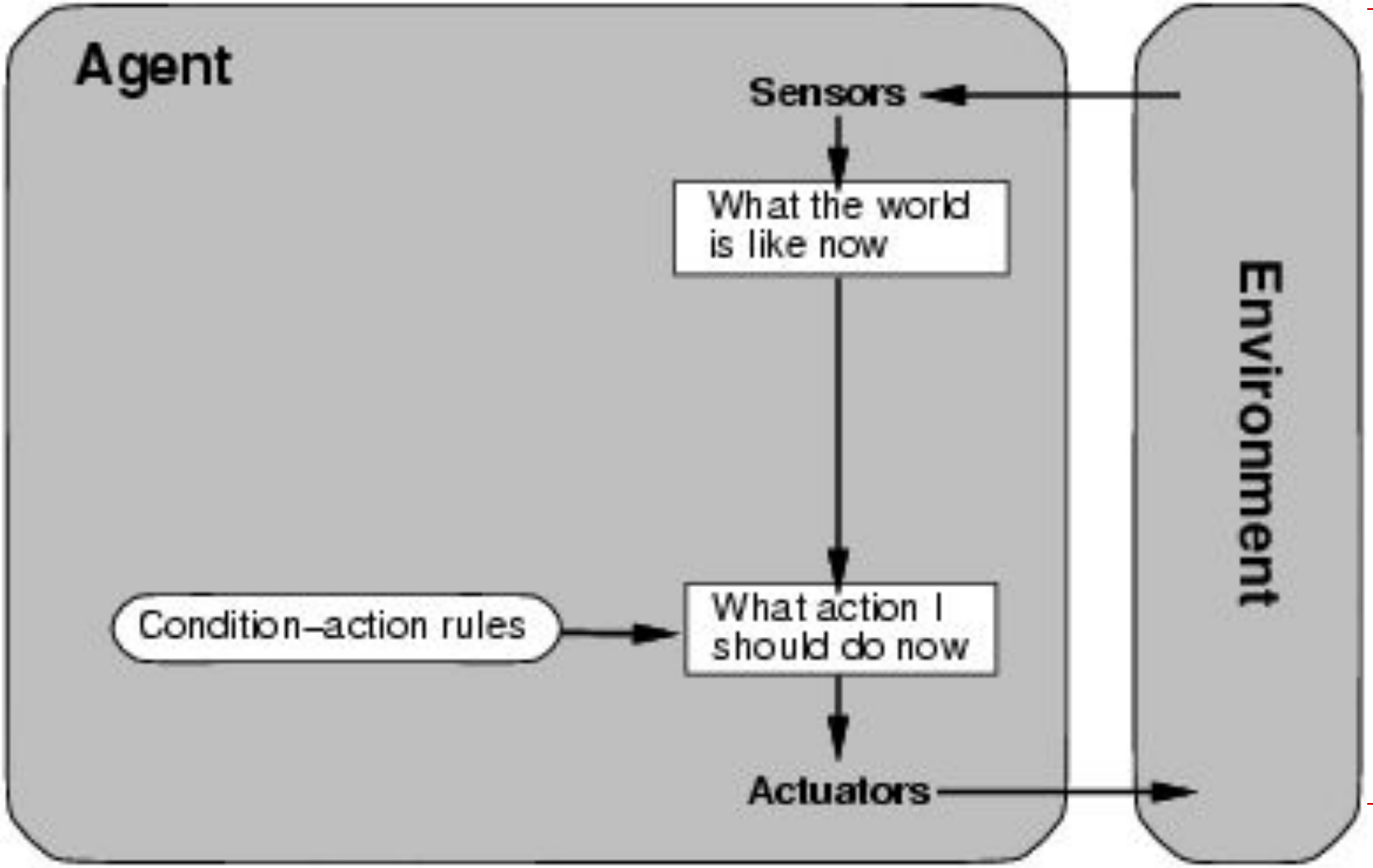
	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

- The environment type largely determines the agent design
- The real world is partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Agent types

- Four basic types
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

Simple reflex agents



Simple reflex agents

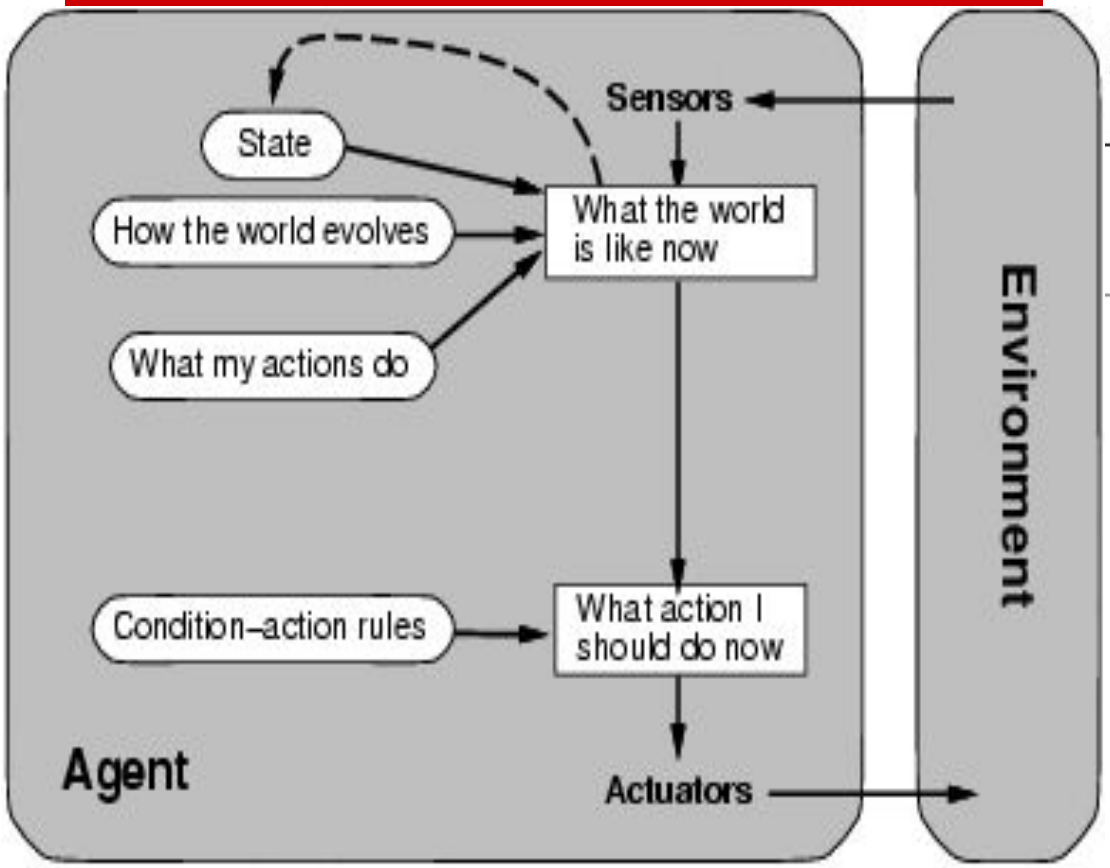
```
function SIMPLE-REFLEX-AGENT(percept) returns an action
  static: rules, a set of condition–action rules

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

- Problem-it will work if the correct decision can be made on the basis of only the current percept- i.e. only if the environment is fully observable

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.
- 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.
- Updating internal state information requires
  - Some information about how world evolves independently of the agent
  - Some information about agent's own action affect the world.
- e.g. overtaking car

Model-based reflex agents



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-STATE(*state*, *action*, *percept*)

rule ← RULE-MATCH(*state*, *rules*)

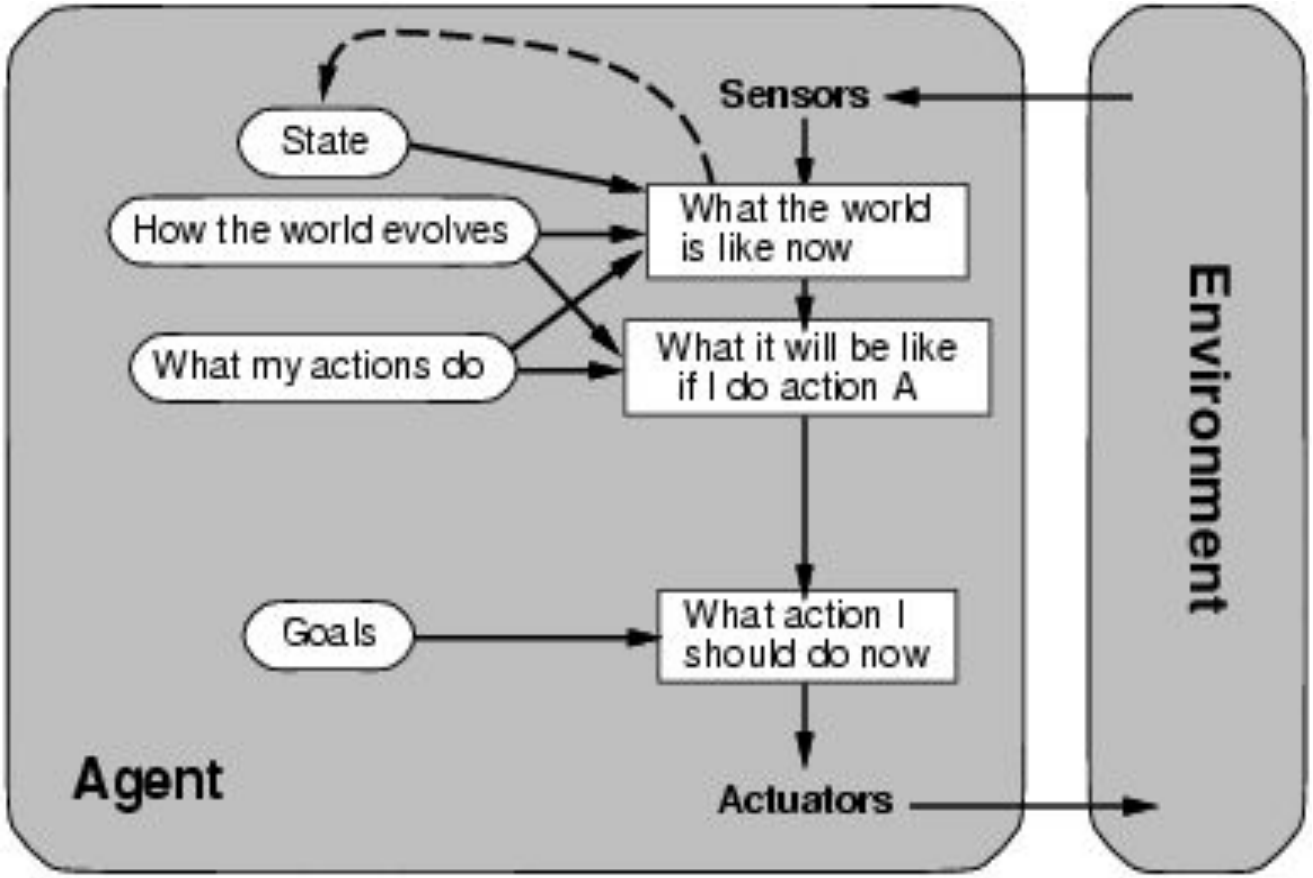
action ← RULE-ACTION[*rule*]

return *action*

Goal Based agents

- Knowing about the current state of the environment is not always enough to decide what to do
- e.g. taxi at road junction
- As well as current state description the agent needs some sort of goal information that describes situation that are desirable

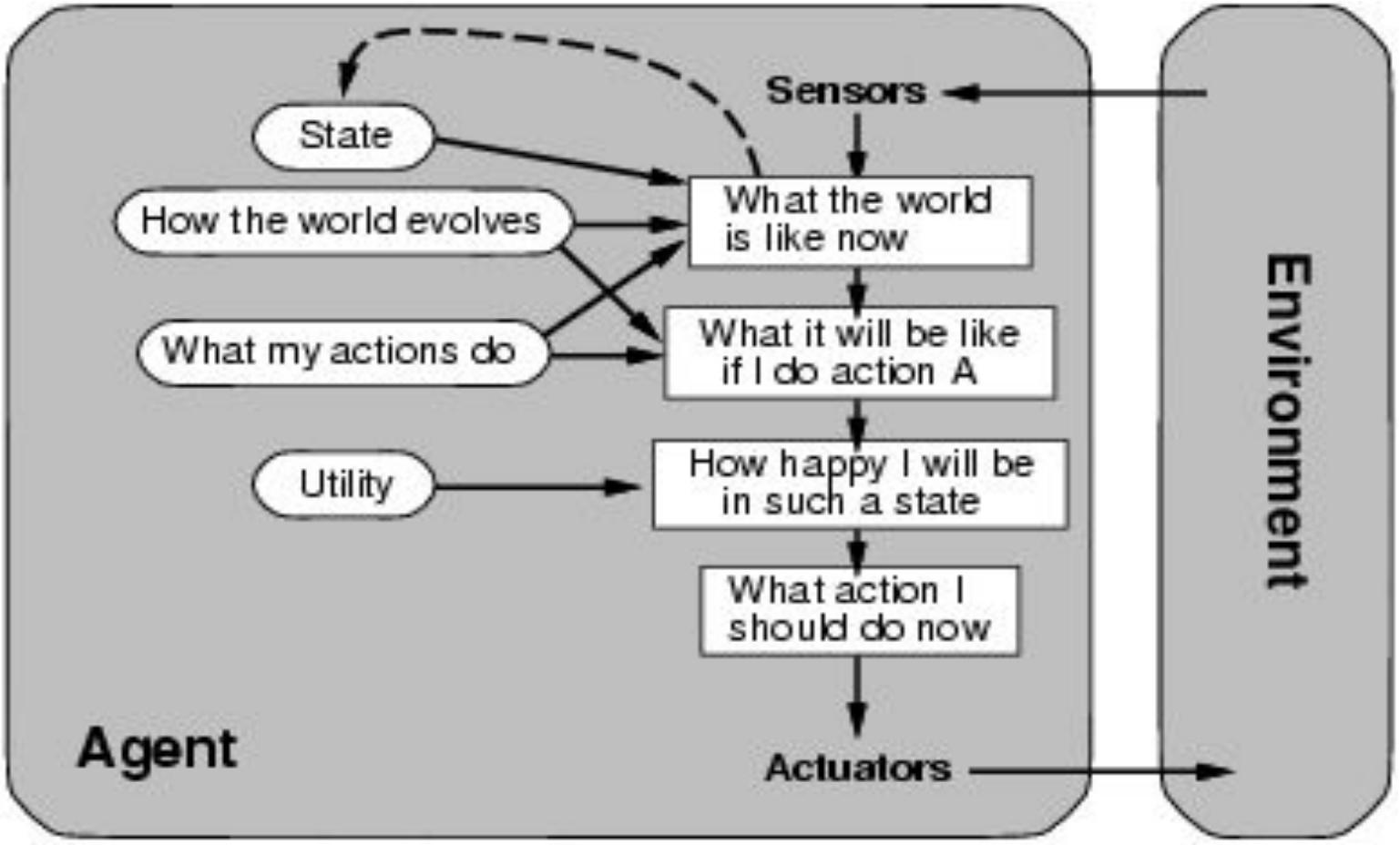
Goal-based agents



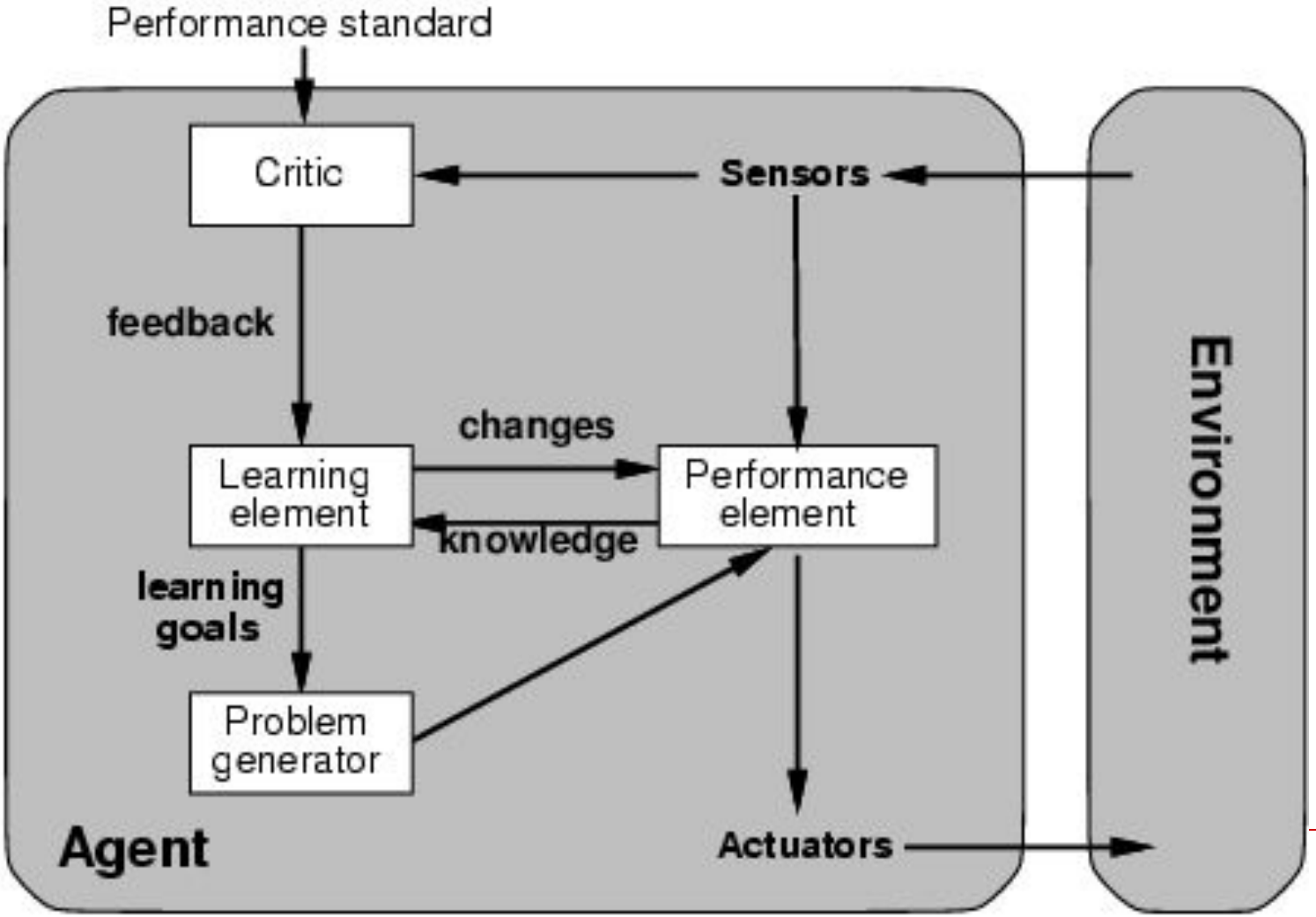
Utility-based agents

- Utility- the quality of being useful
- Utility maps a state or sequence of states onto a real number which describes the associated degree of happiness
- Utility Function allows rational decision
  - In case of conflicting goals utility function provides tradeoff
  - When there are several goals that the agent can aim for, none of them can be achieved with certainty , utility provides likelihood of success can weighted up against the importance of the goals

Utility-based agents



Learning agents



Learning agents

- Learning element
  - Is responsible for making improvements
  - It takes feedback from critic on how the agent is doing and determines how the performance element should be modified to do better in the future

- Performance element
  - Is responsible for selecting external actions
  - It takes percepts and decides the actions
  - Same like agent

Critic

- It tells learning element how well the agent is doing with respect to a fixed performance standard
- Critic is necessary because the percept themselves provide no indication of the agent's success
  e.g. chess – checkmate

Problem generator

- It is responsible suggesting actions that will lead to new and informative experiences
- Its job is to suggest exploratory actions
  e.g. scientists

Summary

- Agents interact with environments through actuators and sensors
- The agent function describes what the agent does in all circumstances
- The performance measure evaluates the environment sequence
- A perfectly rational agent maximizes expected performance
- Agent programs implement (some) agent functions
- PEAS descriptions define task environments
- Environments are categorized along several dimensions:
- observable? deterministic? episodic? static? discrete? single-agent?
- Several basic agent architectures exist: simple reflex, reflex with state, goal-based, utility-based