# Intelligent Agents[1]

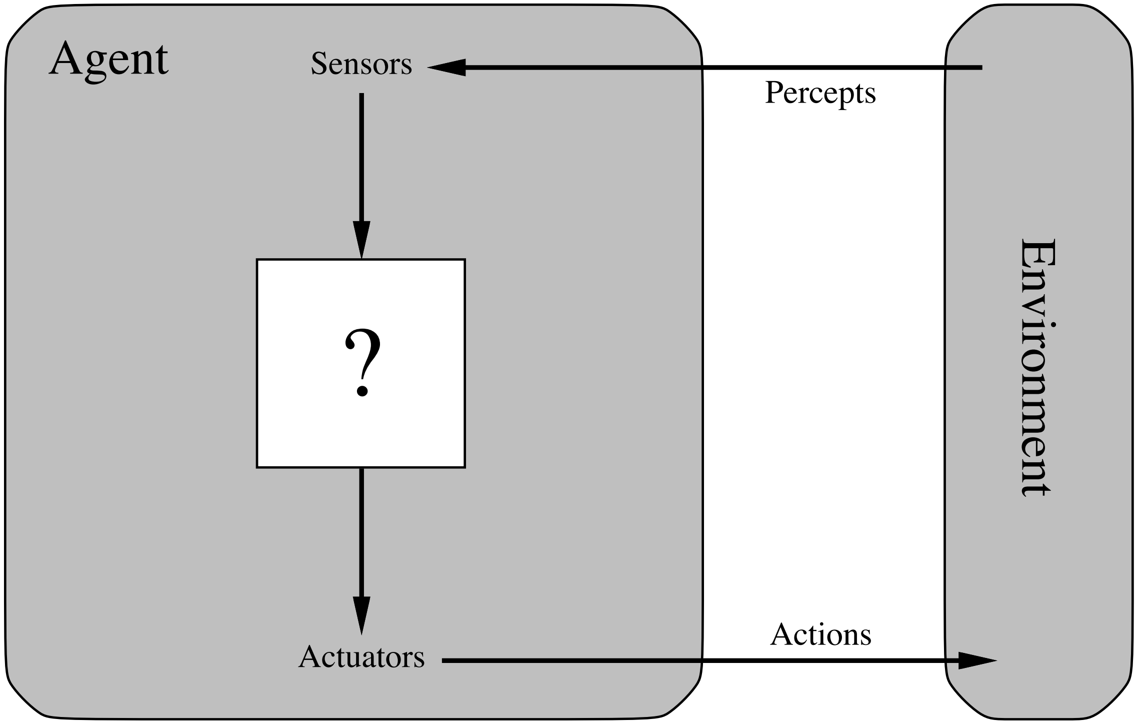
## Lecture 2

[1] The slides have been prepared using the textbook material available on the web, and the slides of the previous editions of the course by Prof. Luigia Carlucci Aiello

# Summary

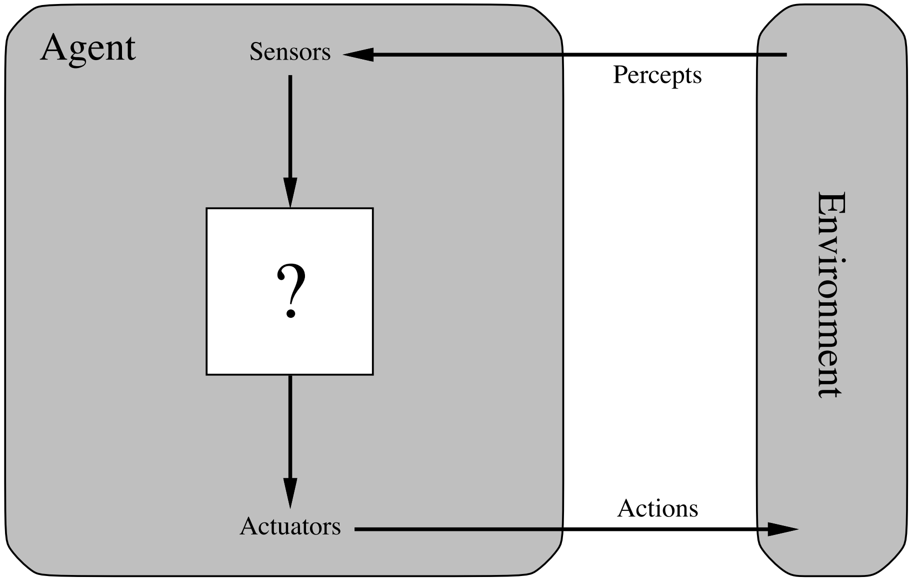
- ♢ Agents and Environments
- ♢ Functions and Programs for Agents
- ♢ Environment specification
- ♢ Environment types
- ♢ Agent types

# Agent and Environment



Agents include humans, robots, softbots, thermostats, etc.

# Agent architecture

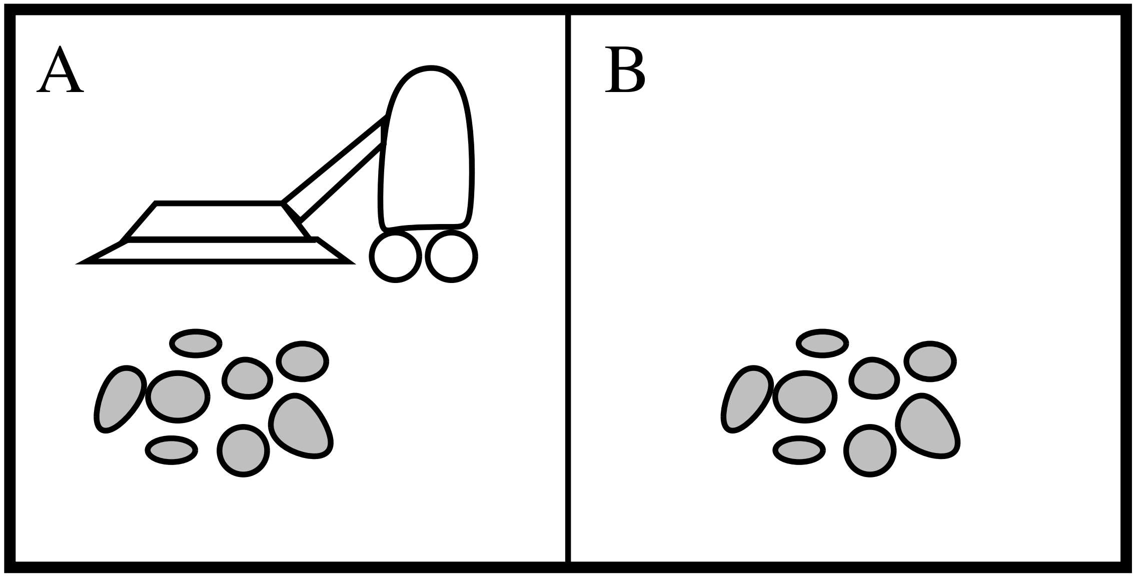


The agent function maps from percept histories to actions:

$$f : \mathcal{P}^* \rightarrow \mathcal{A}$$

The agent program runs on a physical architecture to produce $f$

# Vacuum-cleaner world



Percepts: location and contents, e.g., $[A, Dirty]$

Actions: $Left$, $Right$, $Suck$, $NoOp$

# Rationality

Fixed performance measure evaluates the environment sequence

- one point per square cleaned up in time $T$?
- one point per clean square per time step, minus one per move?
- penalize for $> k$ dirty squares?

A rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date

Rational $\neq$ omniscient
Rational $\neq$ clairvoyant
Rational $\neq$ successful

Rational $\Rightarrow$ learning, exploration (autonomy)

# Functions and agent programs

Agent specification: agent function which maps sequences of percepts into actions

Idealistic implementation: table; Pratical implementation: a program, that may also keep track of the sequence of input percepts

```
function SKELETON-AGENT( percept) returns action
    static: memory, agent's memory of the world
    memory ← UPDATE-MEMORY(memory, percept)
    action ← CHOOSE-BEST-ACTION(memory)
    memory ← UPDATE-MEMORY(memory, action)
    return action
```

# PEAS

To design a rational agent, we must specify the task environment

Consider, e.g., the task of designing an automated taxi:

Performance measure?? safety, destination, profits, legality, comfort, . . .

Environment?? US streets/freeways, traffic, pedestrians, weather, . . .

Actuators?? steering, accelerator, brake, horn, speaker/display, . . .

Sensors?? video, accelerometers, gauges, engine sensors, keyboard, GPS, . . .

# Soccer Robot

Performance??

Environment??

Actions??

Sensors??

# Environment’s Features

- ♢ Observable (Partially)
- ♢ Deterministic (non-deterministic, stochastic)
- ♢ Episodic (Sequential)
- ♢ Static (Dynamic, Semidynamic)
- ♢ Discrete (Continuous)
- ♢ Single Agent (Multi)

The environment influences the agent design

## Environment types

| | Soli | Backgam | Image | Ishop | Taxi |
|---|---|---|---|---|---|
| Observable?? | Yes | Yes | Yes | No | No |
| Deterministic?? | Yes | No | Yes | Partly | No |
| Episodic?? | No | No | Yes | No | No |
| Static?? | Yes | Semi | Semi | Semi | No |
| Discrete?? | Yes | Yes | No | Yes | No |
| Single-agent?? | Yes | No | Yes | Yes (*) | No |

The environment type largely determines the agent design

The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

# Vacuum cleaner world

Performance

– +100 for every durt sucked
– -1 for every action
– -1000 to turn off not at home position

Environment

– square grid, with walls and obstacles
– creation and distribution of durt, bag
– motion actions: move the agent when no obstacles
– sucking action: puts durt in the bag

Sensors (`<bump> <durt> <home>`)

Actions `turnoff forward suck (turnleft) (turnright)`

Observable? Deterministic? Episodic? Static? Discrete?

# Environment Simulation

```
procedure Run-Environment(state, Update-Fn, agents, termination)
    inputs: state, the initial state of the environment
            Update-Fn, function to modify the environment
            agents, a set of agents
            termination, a predicate to test when we are done

    repeat
        for each agent in agents do
            Percept[agent] ← Get-Percept(agent, state)
        end
        for each agent in agents do
            Action[agent] ← Program[agent](Percept[agent])
        end
        state ← Update-Fn(actions, agents, state)
    until termination(state)
```

# Agent types

Four basic types in order of increasing generality:

– simple reflex agents

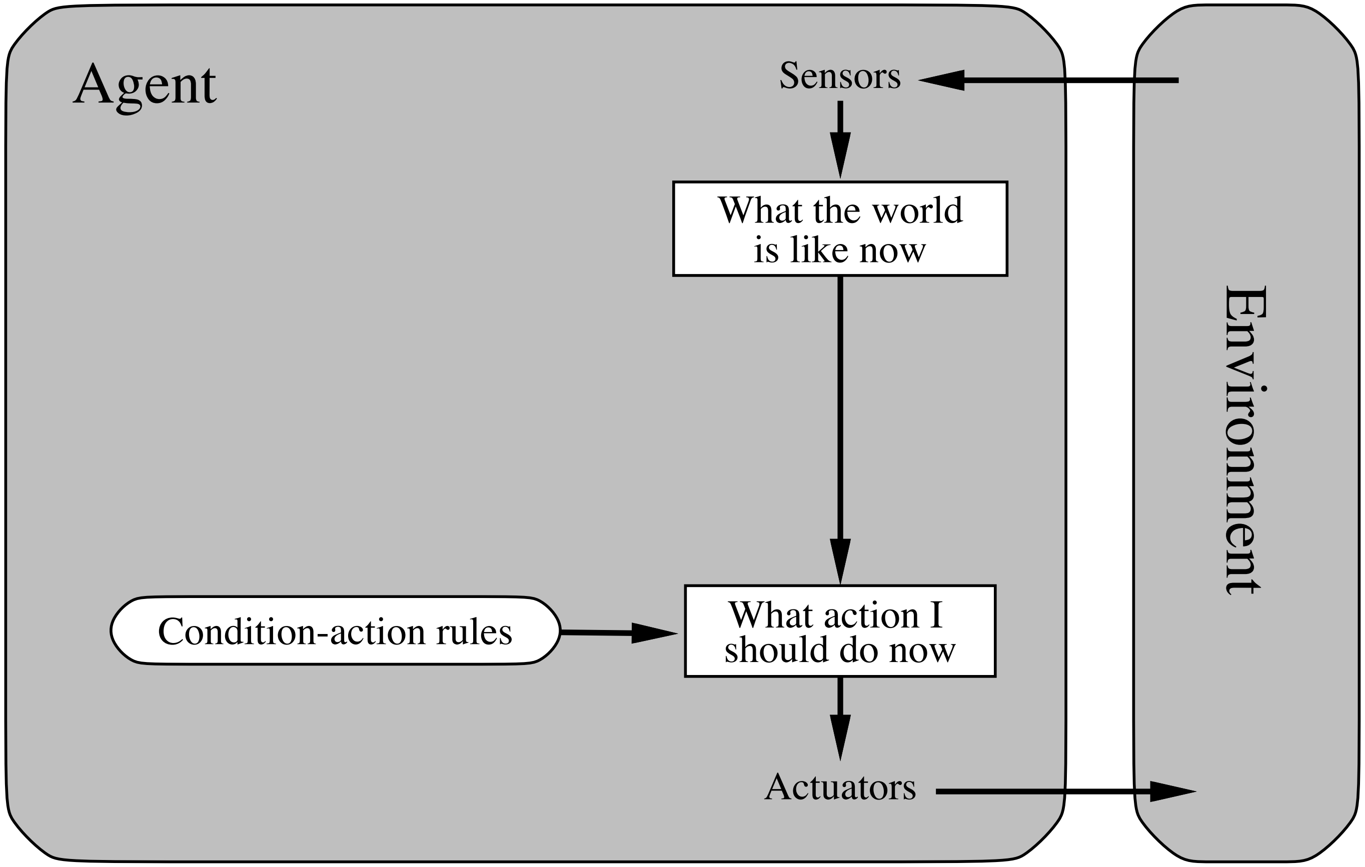
– reflex agents with state

– goal-based agents

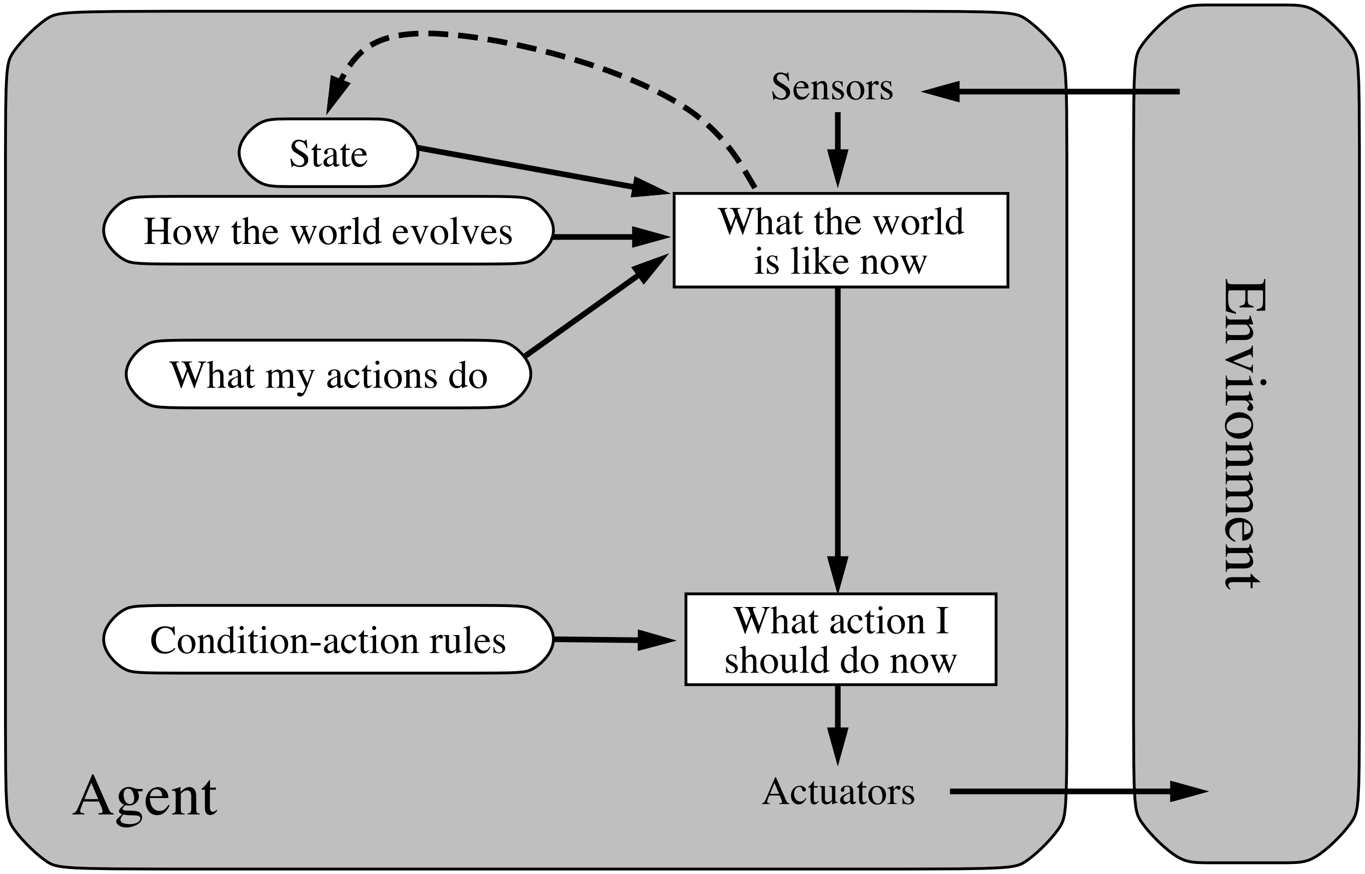
– utility-based agents

All these can be turned into learning agents

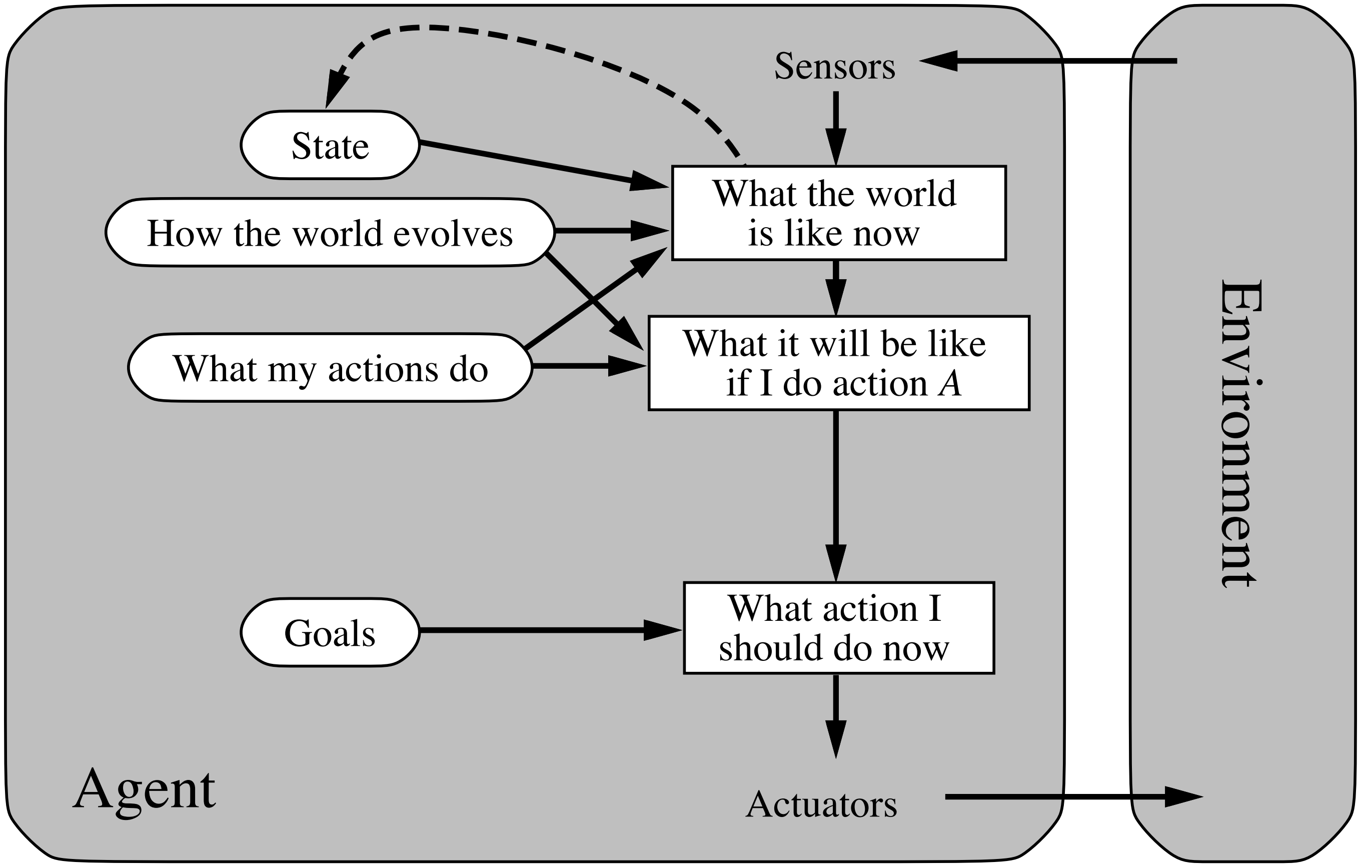
# Simple reflex agents



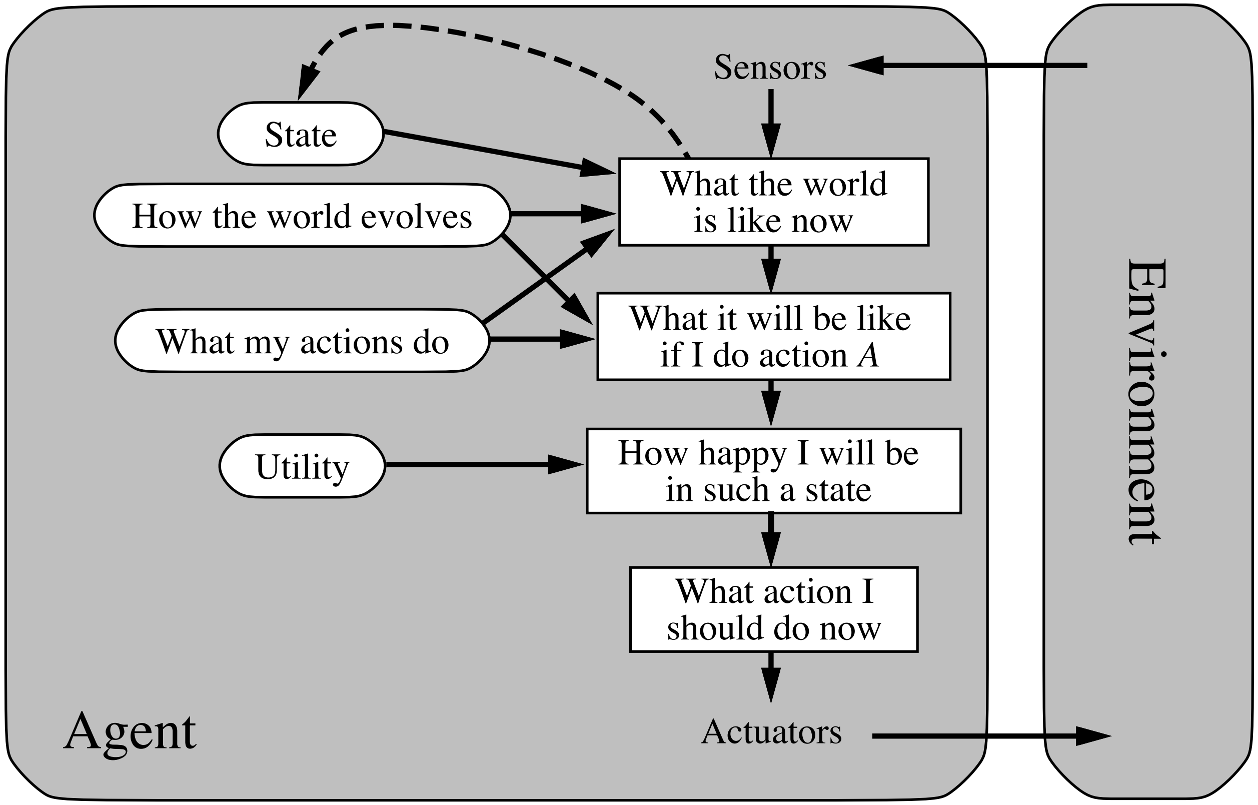
# Reflex agents with state



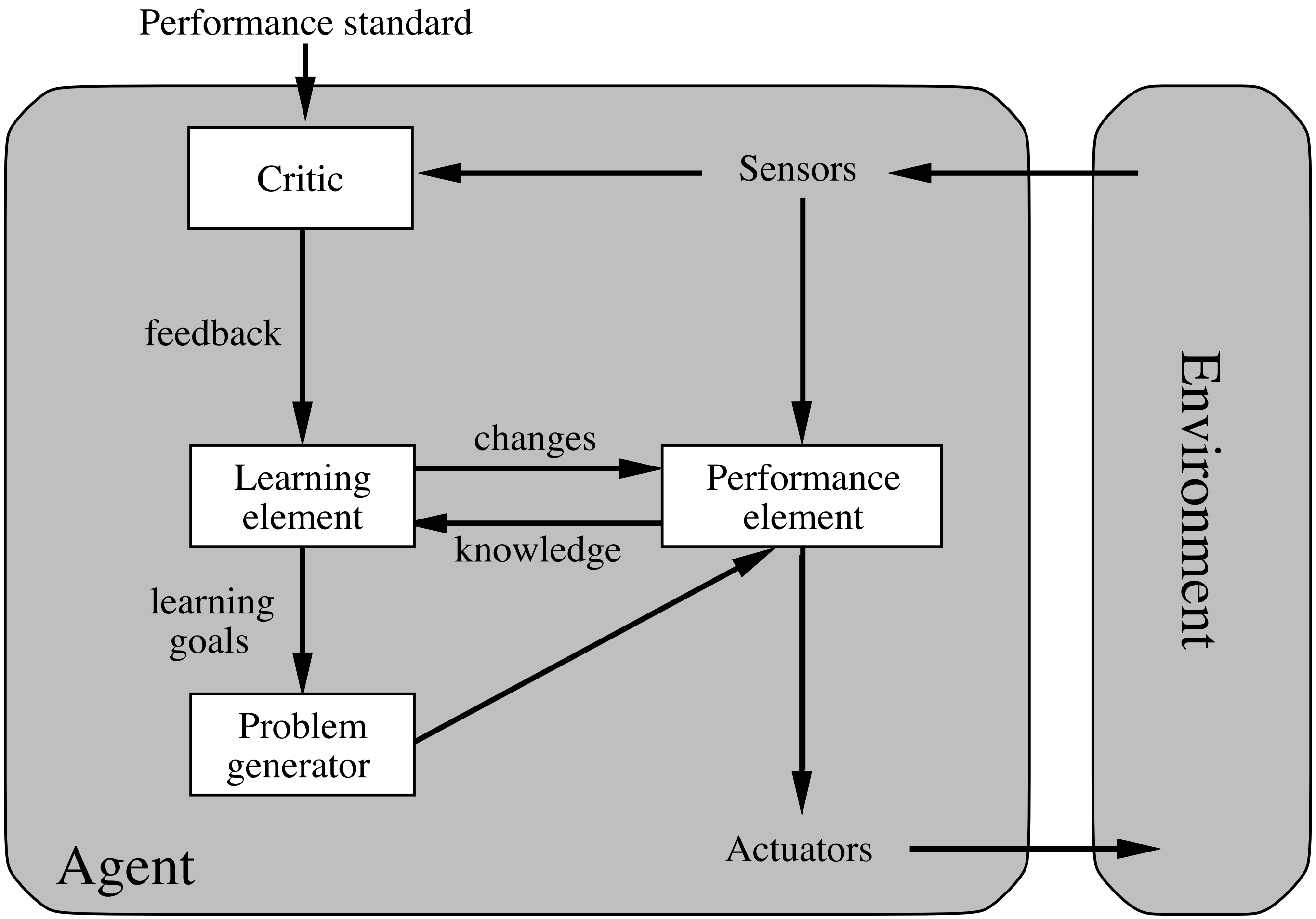
# Goal-based agents



# Utility-based agents


Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action *A*
What my actions do
Utility
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

# Learning agents



# Summarizing

AI agents have the following features:

- ♢ Perception
- ♢ Reasoning
- ♢ Action

Note:

- ♢ applicable robots and softbot
- ♢ integration of all the above
- ♢ a chess player robot should perceive the board, the moves . . .