Intelligent Agents[1]

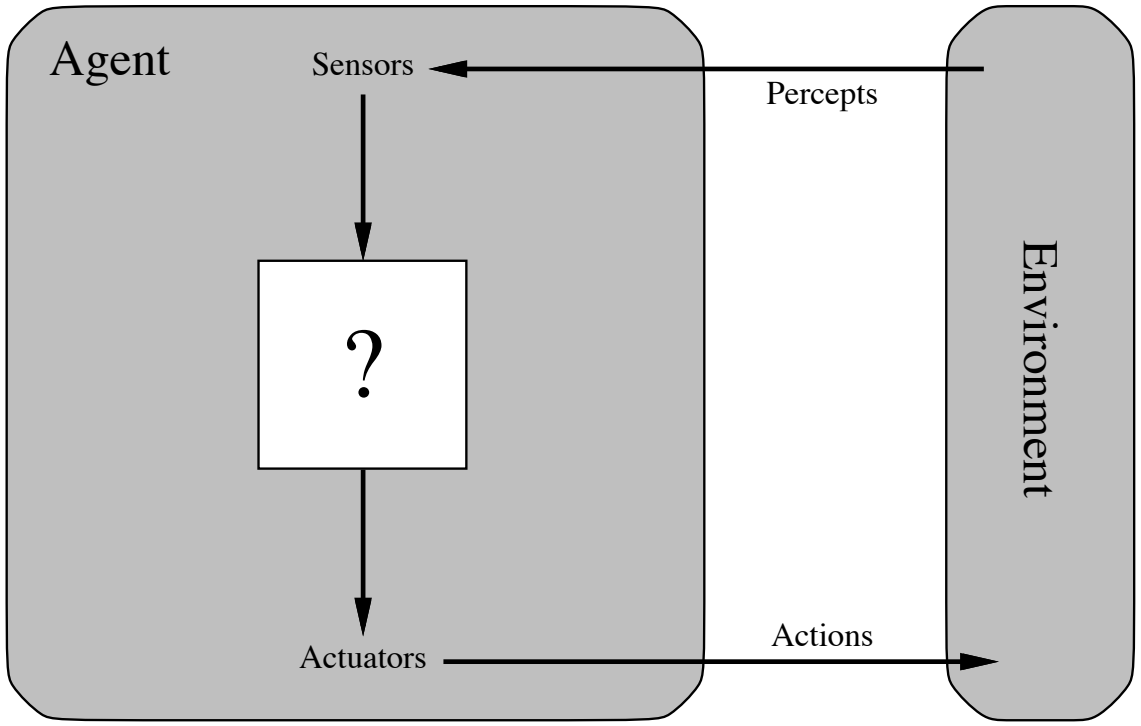
Lecture 2

[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

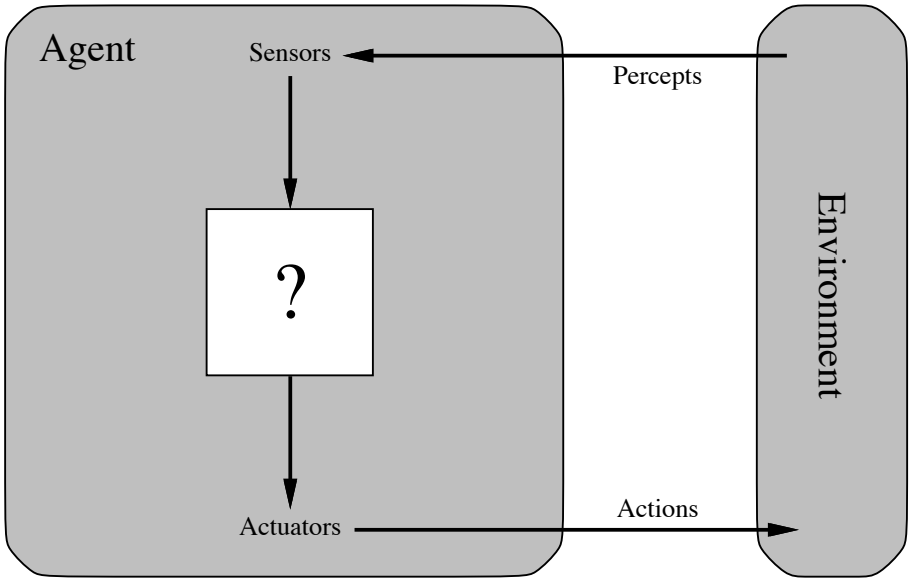
- ♢ Agents and Environments
- ♢ Functions and Programs for Agents
- ♢ Environment specification
- ♢ Environment types
- ♢ Agent types

Agent and Environment



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

Agent architecture

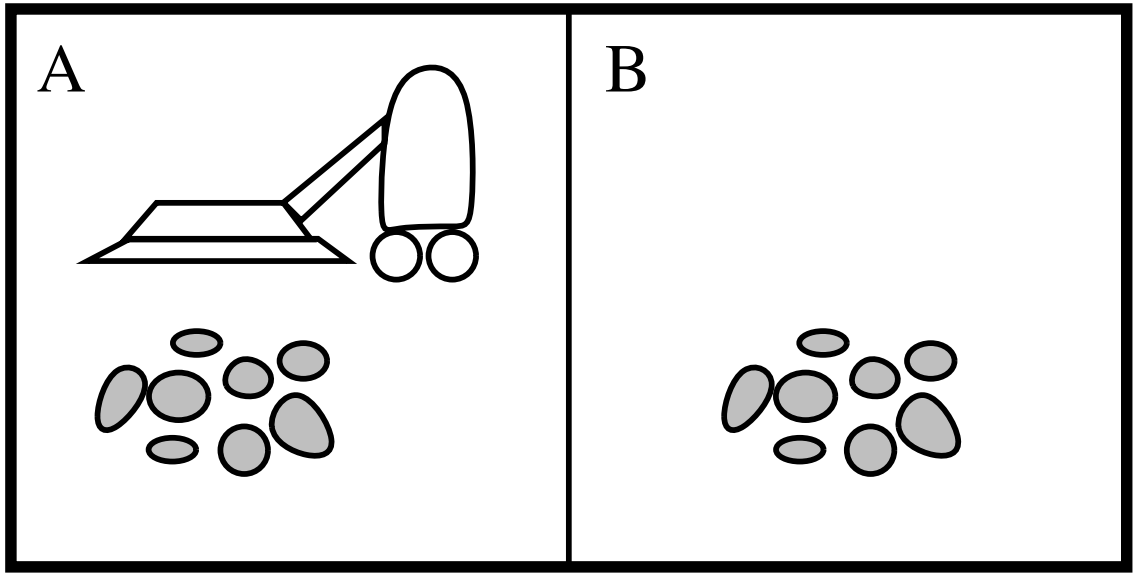


The agent function maps from percept histories to actions:

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

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*

The tabular function for the vacuum cleaner

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

An intelligent agent for the vacuum cleaner

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*

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)

If the next state of the environment is completely determined by the current state and the action executed by the agent(s), then we say the environment is deterministic

If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static.

\+ Known vs Unknown

The environment influences the agent design

Environment types

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

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*)

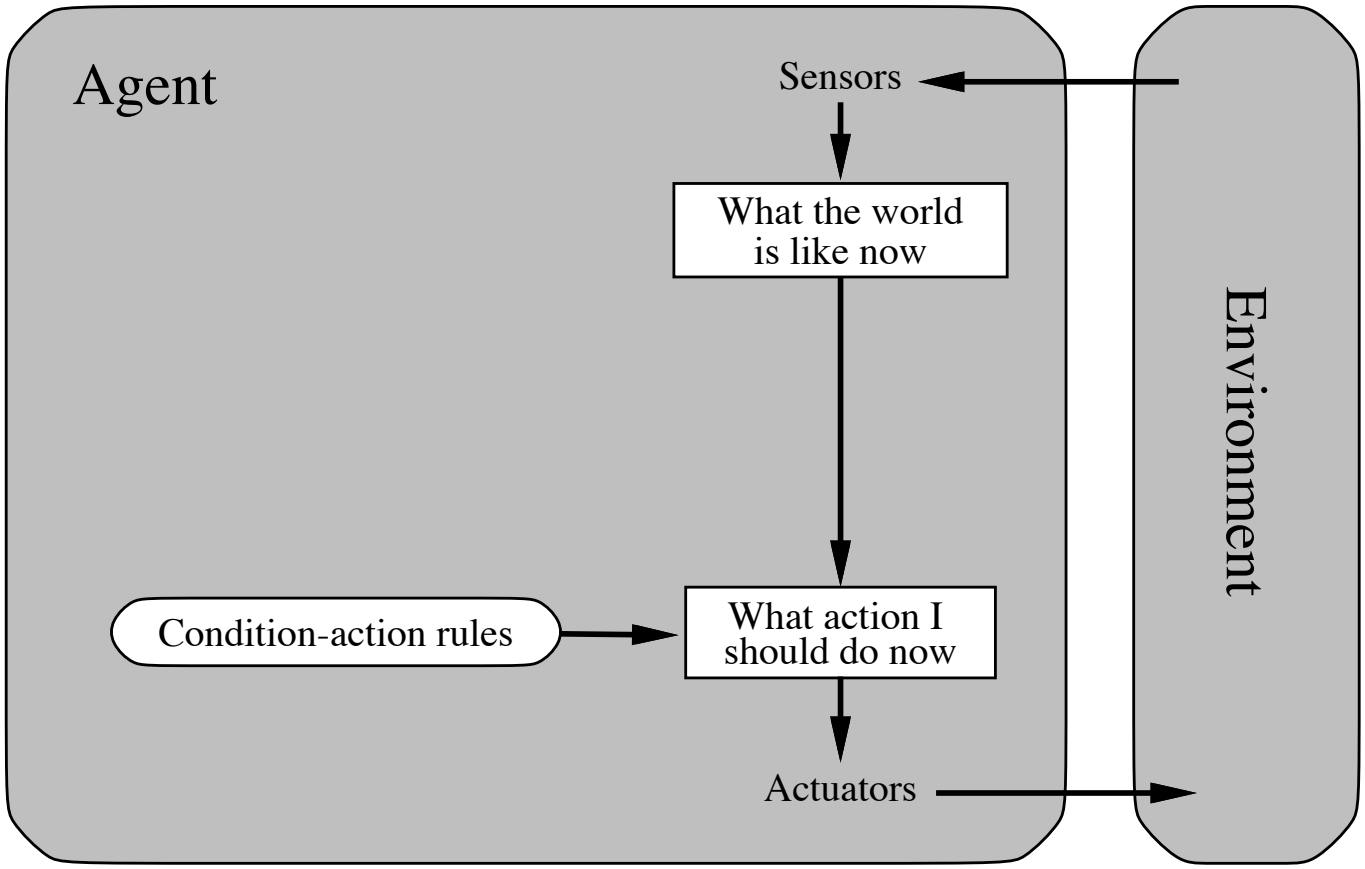
Agent types

Four basic types in order of increasing generality:

- simple reflex agents
- model-based reflex agents
- goal-based agents
- utility-based agents

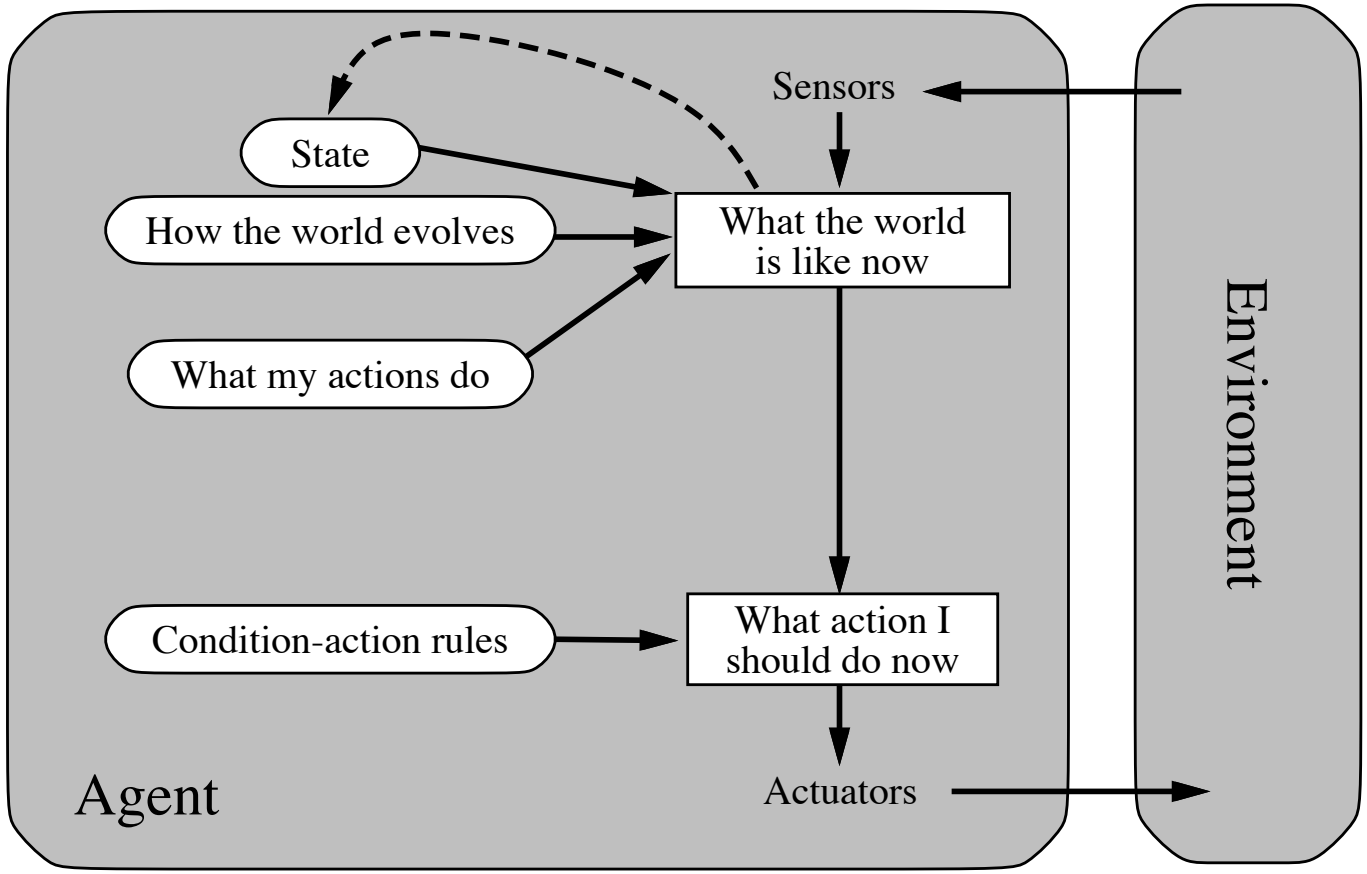
All these can be turned into learning agents

Simple reflex agents



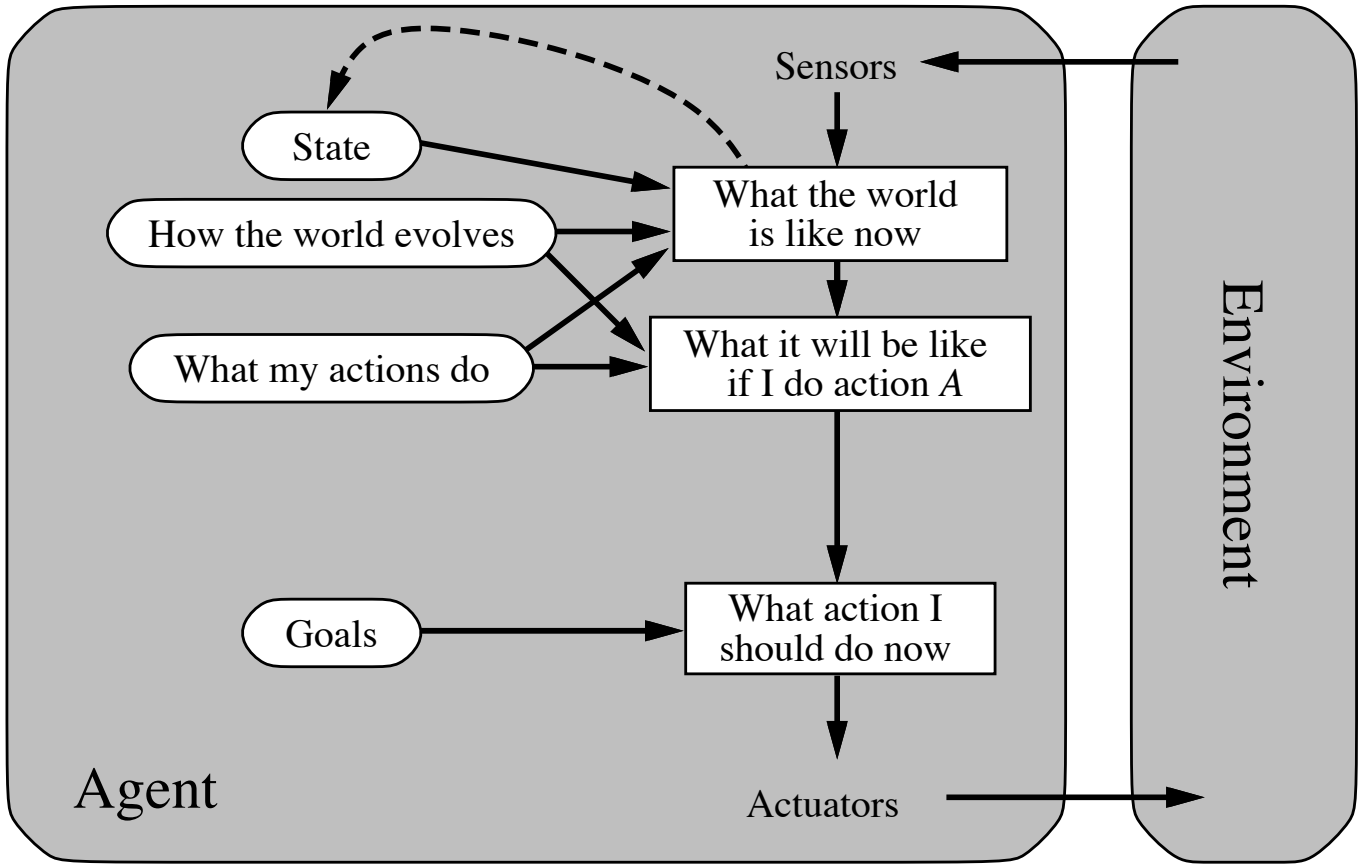
These agents select actions on the basis of the current percept, ignoring the rest of the percept history.

Reflex agents with state

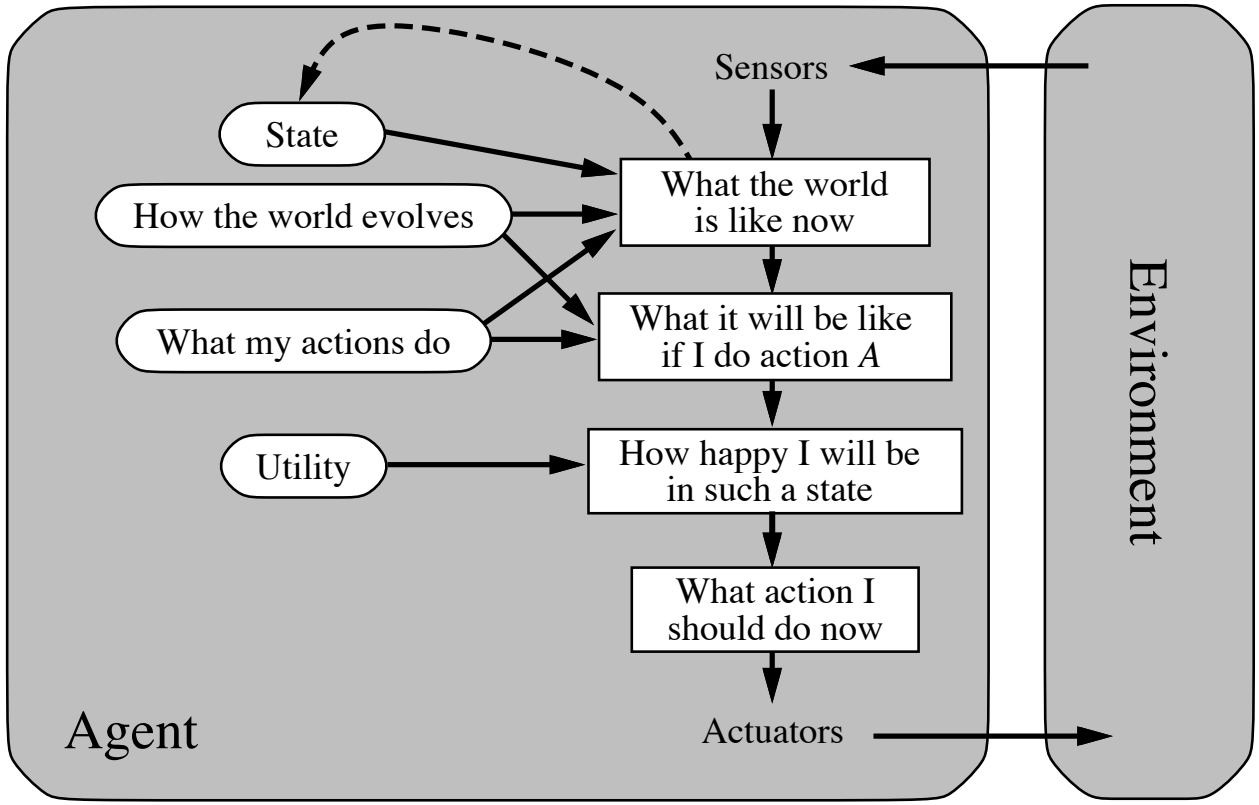


The most effective way to handle partial observability is for the agent to keep track of the part of the world it can't see now. That is, the agent should maintain some sort of internal state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.

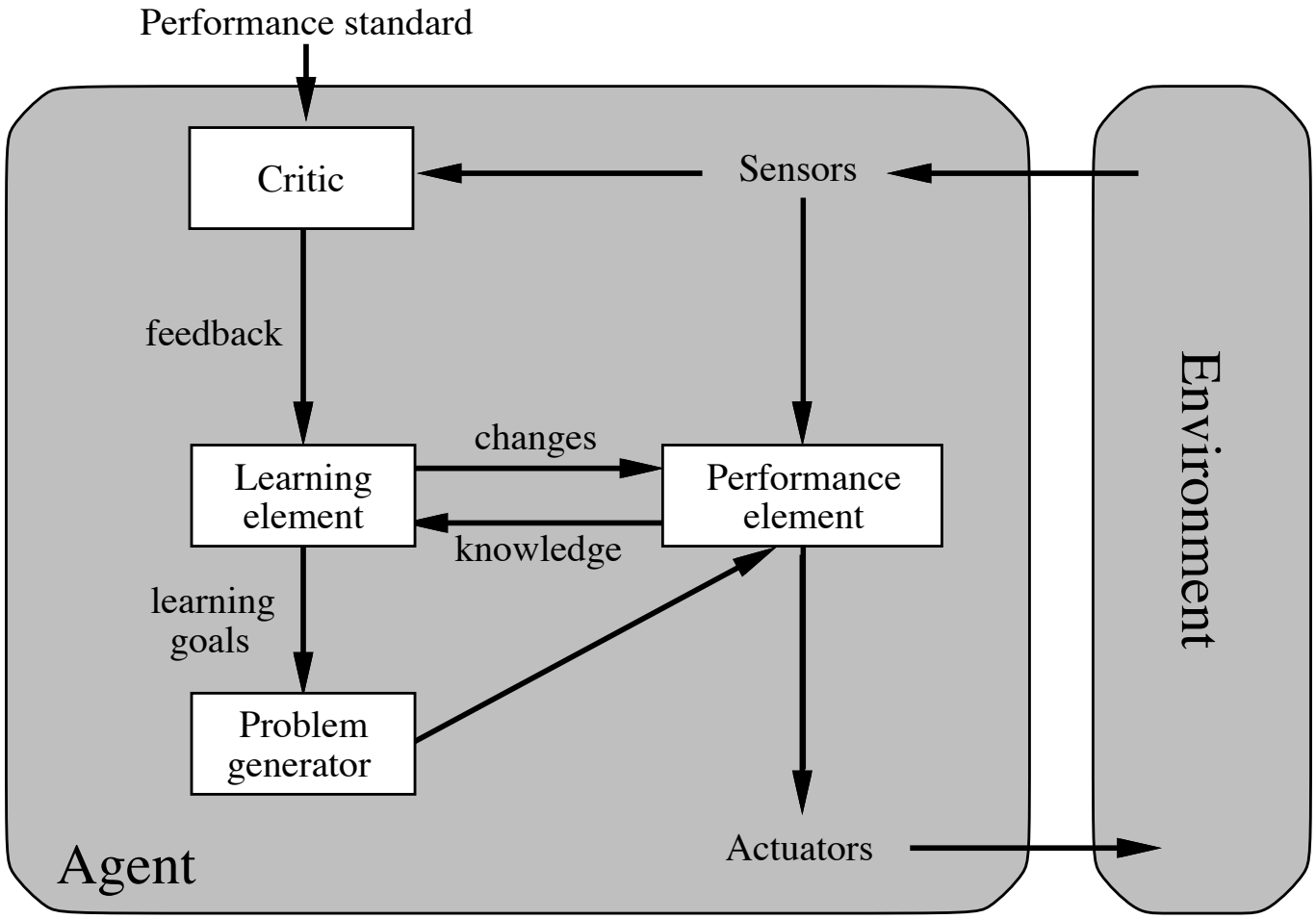
Goal-based agents



Utility-based agents



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 . . .