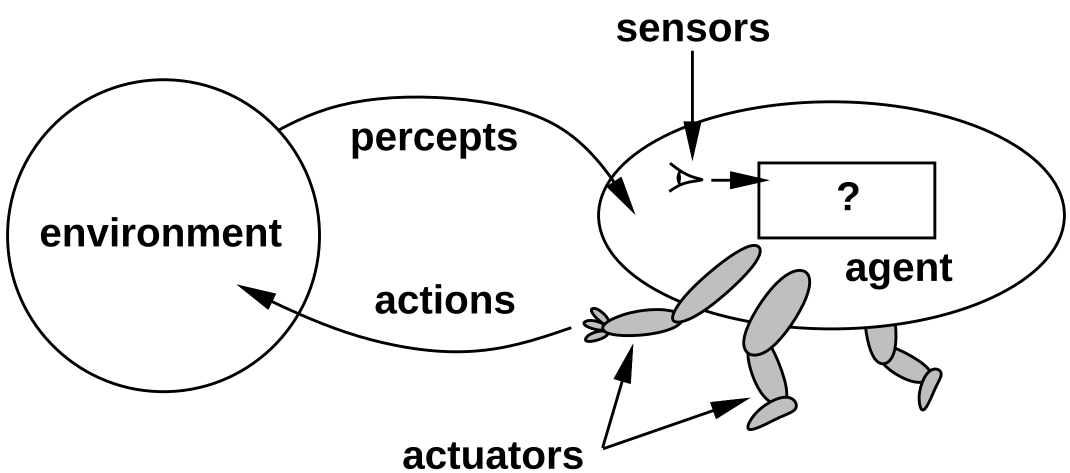
Intelligent Agents

Chapter 2

Outline

- ♢ Agents and environments
- ♢ Rationality
- ♢ PEAS (Performance measure, Environment, Actuators, Sensors)
- ♢ Environment types
- ♢ Agent types

Agents and environments



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

The agent function maps from percept histories to actions:

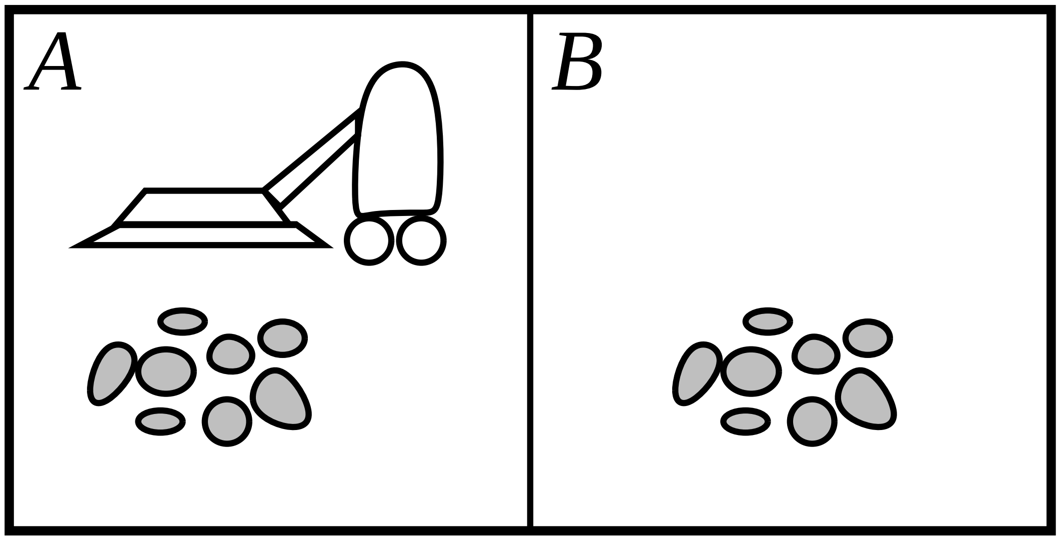
$$f : \mathcal{P}^* \rightarrow \mathcal{A}$$

The agent program runs on the physical architecture to produce f

An agent can perceive its own actions, but not always it effects.

The agent function will internally be represented by the agent program.

Vacuum-cleaner world



Environment: square A and B.

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

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

A vacuum-cleaner agent

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

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*

What is the **right** function?
Can it be implemented in a small agent program?

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

– percepts may not supply all relevant information

Rational $\neq$ clairvoyant

– action outcomes may not be as expected

Hence, rational $\neq$ successful

Rationality

Rational $\Rightarrow$ exploration, learning, autonomy

The proposed definition of rationality requires

– Information gathering/exploration

– – To maximize future rewards

– Learn from percepts

– – Extending prior knowledge

– Agent autonomy

– – Compensate for incorrect prior knowledge

PEAS

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

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

Performance measure??

Environment??

Actuators??

Sensors??

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, . . .

Internet shopping agent

Performance measure??

Environment??

Actuators??

Sensors??

Internet shopping agent

Performance measure?? price, quality, appropriateness, efficiency

Environment?? current and future WWW sites, vendors, shippers

Actuators?? display to user, follow URL, fill in form

Sensors?? HTML pages (text, graphics, scripts)

Environment types

Fully vs. partially observable: an environment is full observable when the sensors can detect all aspects that are relevant to the choice of action.

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??				
Deterministic??				
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

Deterministic vs. stochastic: if the next environment state is completely determined by the current state the executed action then the environment is deterministic.

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??				
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

Episodic vs. sequential: In an episodic environment the agents experience can be divided into atomic steps where the agents perceives and then performs A single action. The choice of action depends only on the episode itself

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??				
Static??				
Discrete??				
Single-agent??				

Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??				
Discrete??				
Single-agent??				

Environment types

Static vs. dynamic: If the environment can change while the agent is choosing an action, the environment is dynamic. Semi-dynamic if the agents performance changes even when the environment remains the same.

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??	Yes	Semi	Semi	No
Discrete??				
Single-agent??				

Environment types

Discrete vs. continuous: This distinction can be applied to the state of the environment, the way time is handled and to the percepts/actions of the agent

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??	Yes	Semi	Semi	No
Discrete??	Yes	Yes	Yes	No
Single-agent??				

Environment types

Single vs. multi-agent: Does the environment contain other agents who are also maximizing some performance measure that depends on the current agents actions?

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??	Yes	Semi	Semi	No
Discrete??	Yes	Yes	Yes	No
Single-agent??	Yes	No	Yes (except auctions)	No

Agent types

The environment type largely determines the agent design

The simplest environment is:
– Fully observable, deterministic, episodic, static, discrete and single-agent.

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

Video game is
– ???

Agent types

Four basic types in order of increasing generality:
– simple reflex agents
– reflex agents with state
– goal-based agents
– utility-based agents

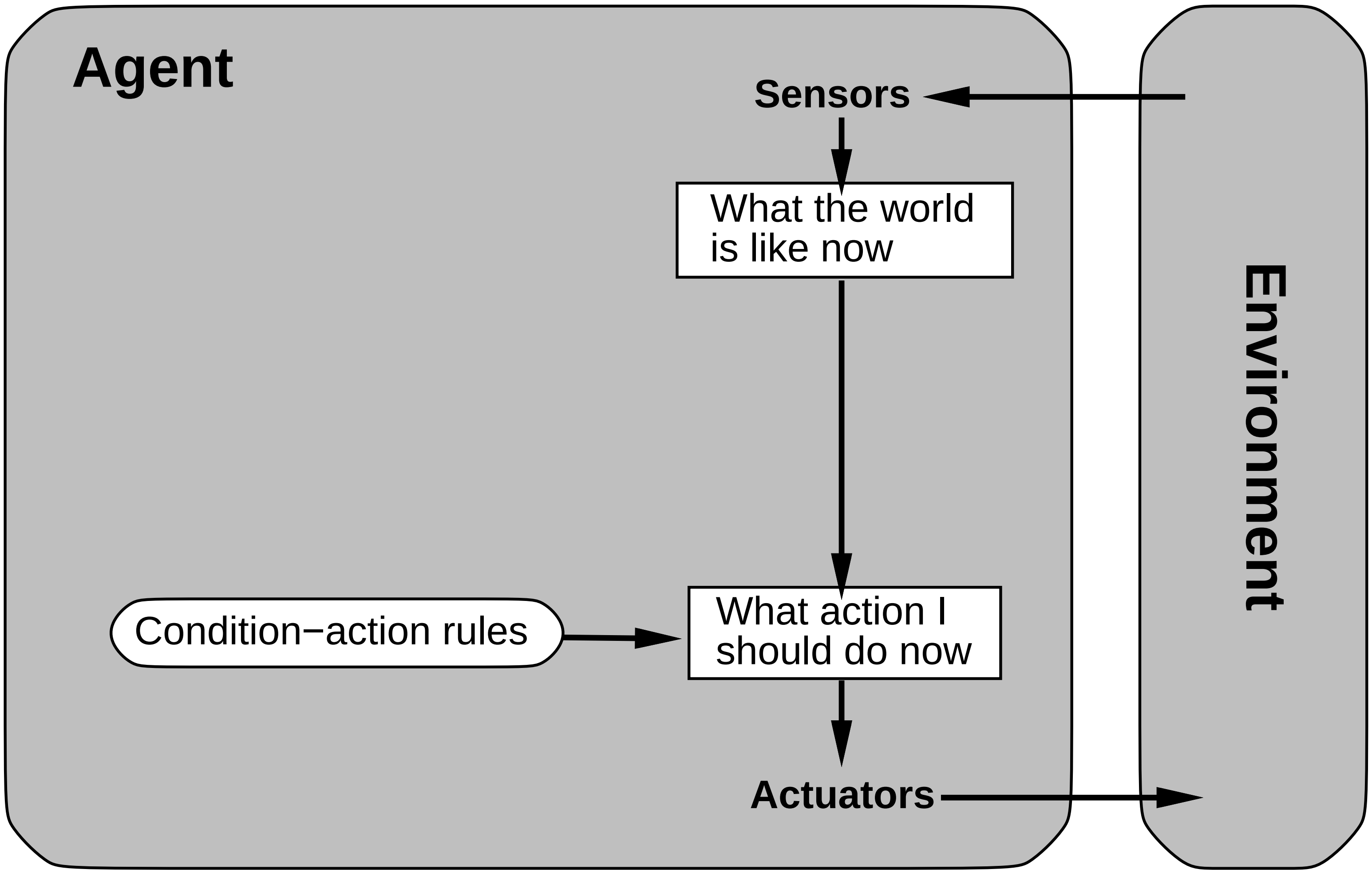
All these can be turned into learning agents

Agen types

How does the inside of the agent work?
– Agent = architecture + program

All agents have the same skeleton:
– Input = current percepts
– Output = action
– Program = manipulates input to produce output

Simple reflex agents



Simple reflex agents

Example

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*

```
(setq joe (make-agent :name 'joe :body (make-agent-body)
                      :program (make-reflex-vacuum-agent-program)))

(defun make-reflex-vacuum-agent-program ()
  #'(lambda (percept)
      (let ((location (first percept)) (status (second percept)))
        (cond ((eq status 'dirty) 'Suck)
              ((eq location 'A) 'Right)
              ((eq location 'B) 'Left)))))
```

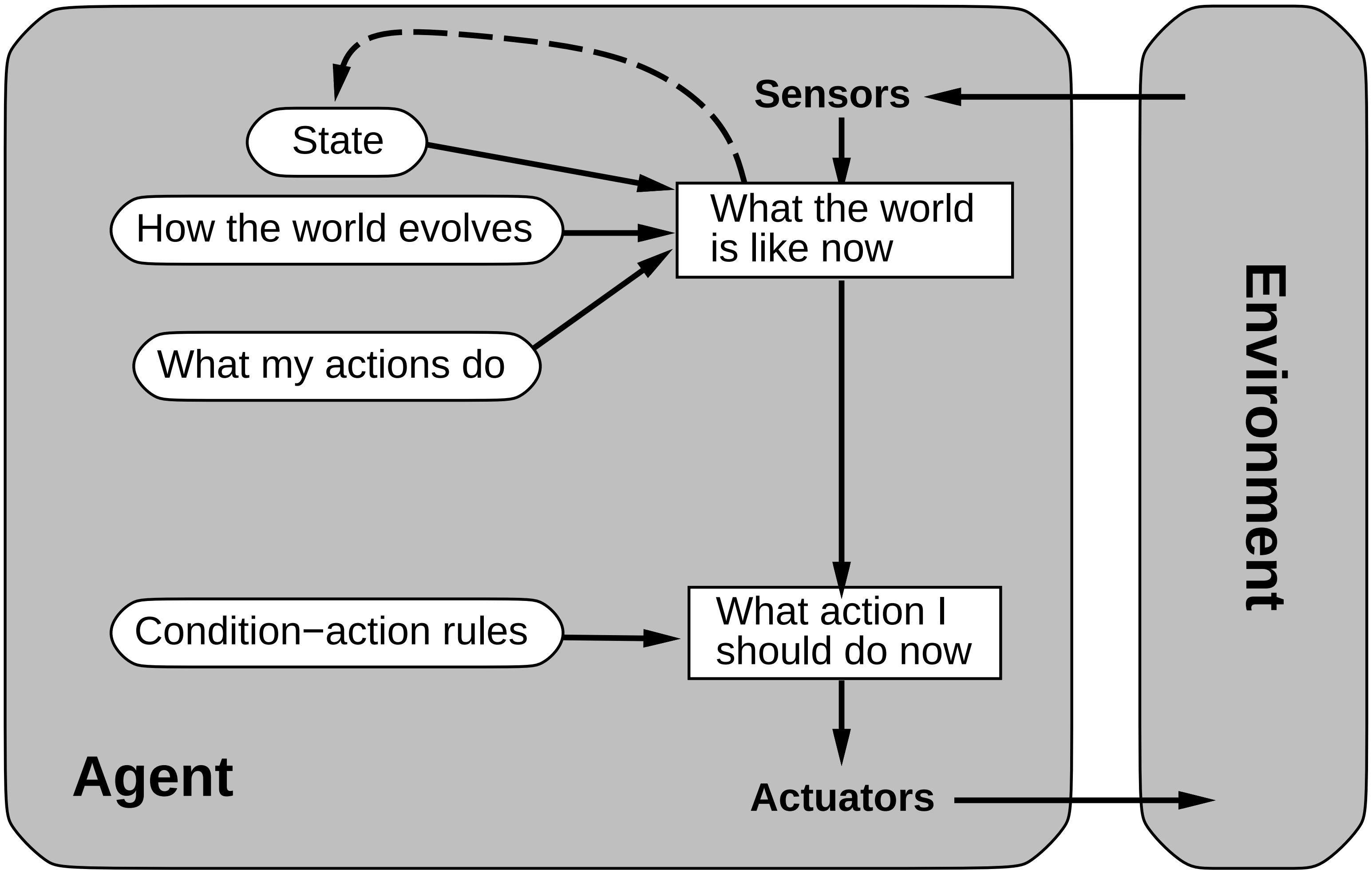
Simple reflex agents

Select action on the basis of only the current percept.

```
function SIMPLE-REFLEX-AGENT(percept) returns an action
    static: rules, a set of condition-action rules
    state = INTERPRET-INPUT(percept)
    rule = RULE-MATCH(state, rule) (this is a simple table look-up step)
    action = RULE-ACTION[rule]
    return action
```

Will only work if the environment is fully observable otherwise infinite loops may occur.

Reflex agents with state



Example

function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action
static: *last_A*, *last_B*, numbers, initially ∞

  if $status = Dirty$ **then** . . .

```
(defun make-reflex-vacuum-agent-with-state-program ()
  (let ((last-A infinity) (last-B infinity))
  #'(lambda (percept)
      (let ((location (first percept)) (status (second percept)))
        (incf last-A) (incf last-B)
        (cond
         ((eq status 'dirty)
          (if (eq location 'A) (setq last-A 0) (setq last-B 0))
          'Suck)
         ((eq location 'A) (if (> last-B 3) 'Right 'NoOp))
         ((eq location 'B) (if (> last-A 3) 'Left 'NoOp)))))))
```

Reflex agents with state

To tackle partially observable environments.

– Maintain internal state

Over time update state using world knowledge

– How does the world change.

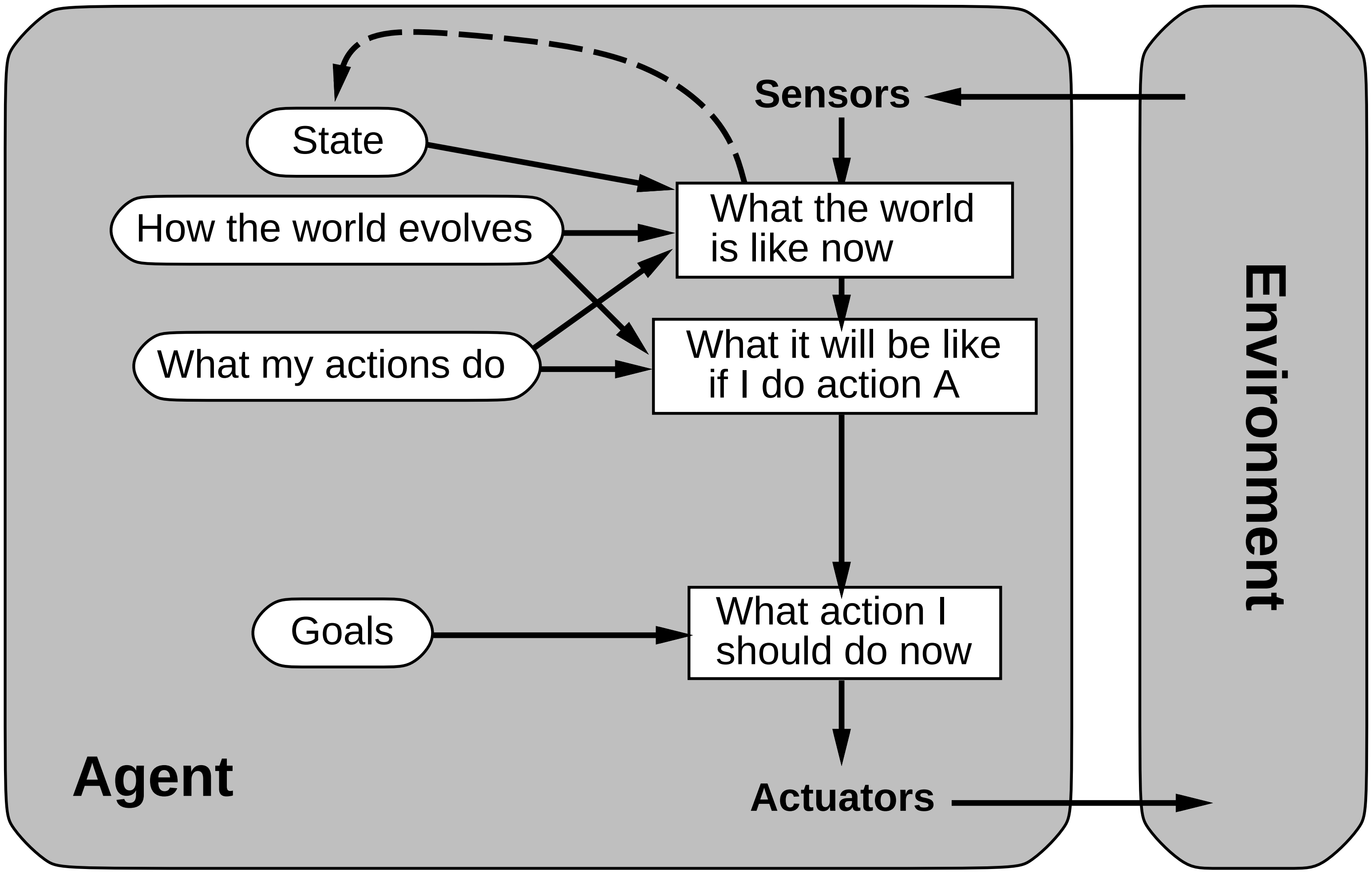
– How do actions affect world.

Need Model of World

Reflex agents with state

function REFLEX-AGENT-WITH-STATE(percept) returns an action
static: rules, a set of condition-action rules state, a description of the current world state action, the most recent action
state = UPDATE-STATE(state, action, percept)
rule =RULE-MATCH(state, rule)
action = RULE-ACTION[rule]
return action

Goal-based agents



Goal-based agents

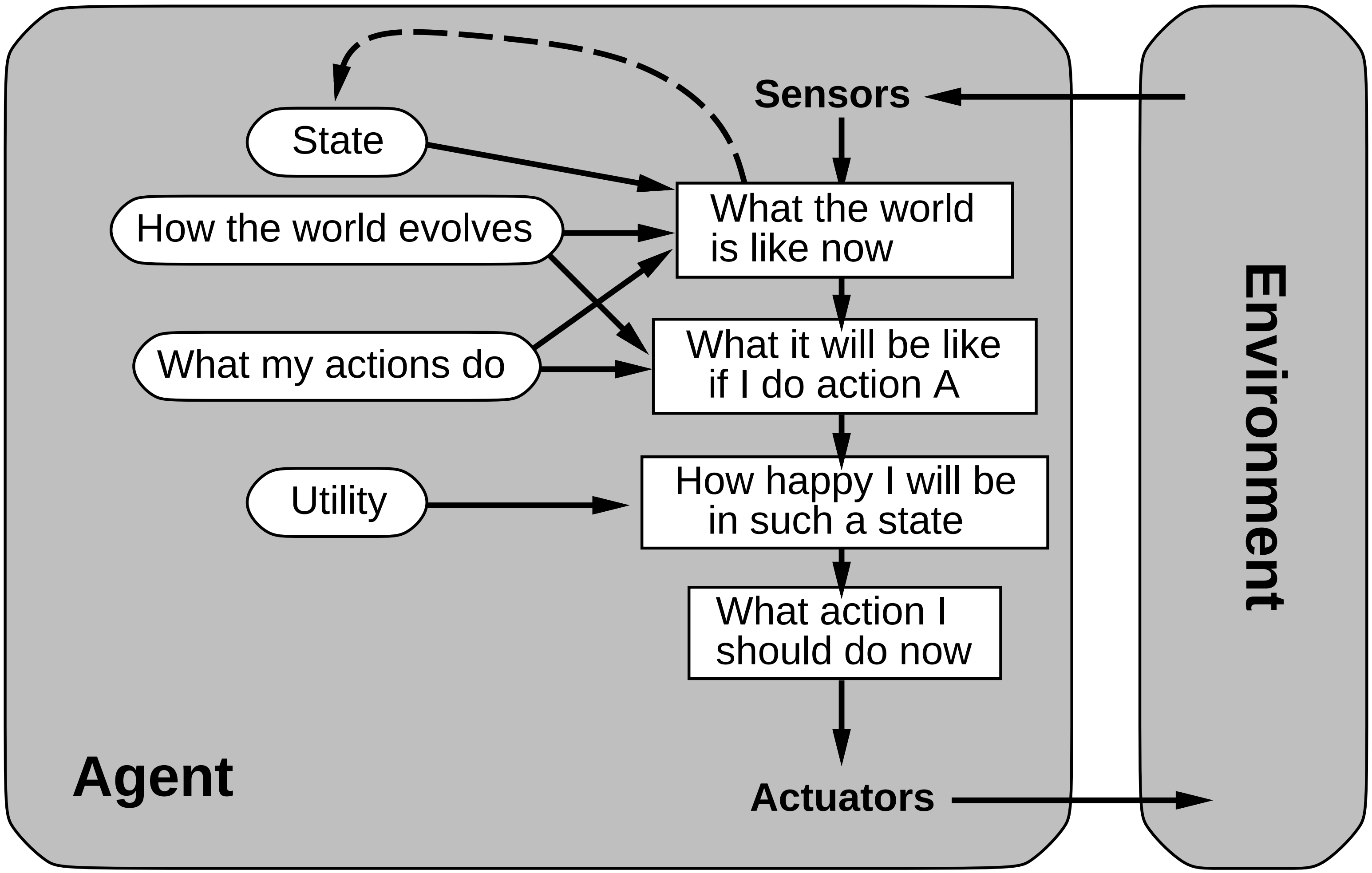
The agent needs a goal to know which situations are desirable.

Things become difficult when long sequences of actions are required to find the goal.

Typically investigated in search and planning research. Major difference: future is taken into account

Is more flexible since knowledge is represented explicitly and can be manipulated.

Utility-based agents



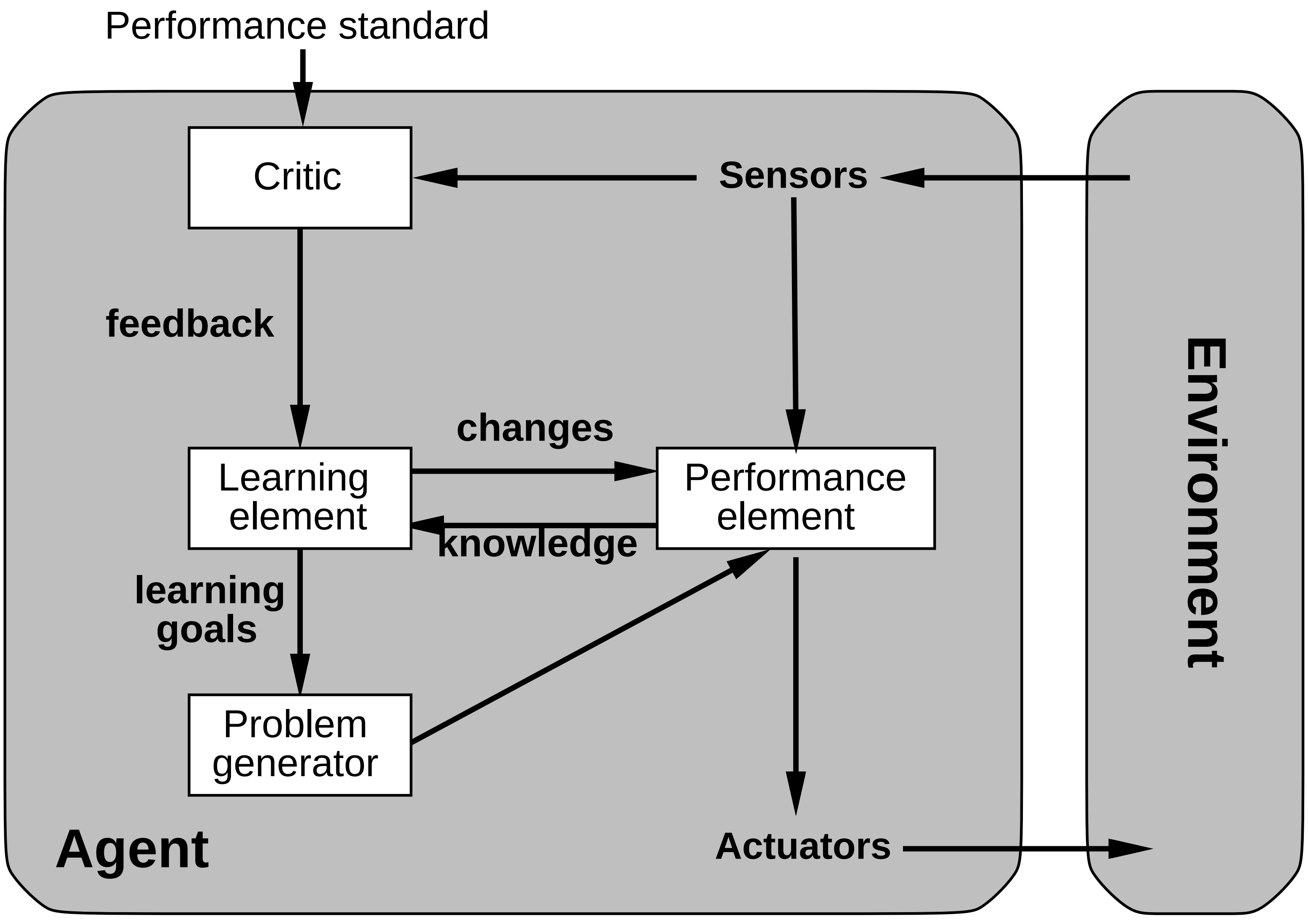
Utility-based agents

Certain goals can be reached in different ways. Some are better, have a higher utility.

Utility function maps a (sequence of) state(s) onto a real number.

Improves on goals:
– Selecting between conflicting goals
– Select appropriately between several goals based on likelihood of success.

Learning agents



Learning agents

All previous agent-programs describe methods for selecting actions. Yet it does not explain the origin of these programs.

Learning mechanisms can be used to perform this task. Teach them instead of instructing them.

Advantage is the robustness of the program toward initially unknown environments.

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:
reflex, reflex with state, goal-based, utility-based