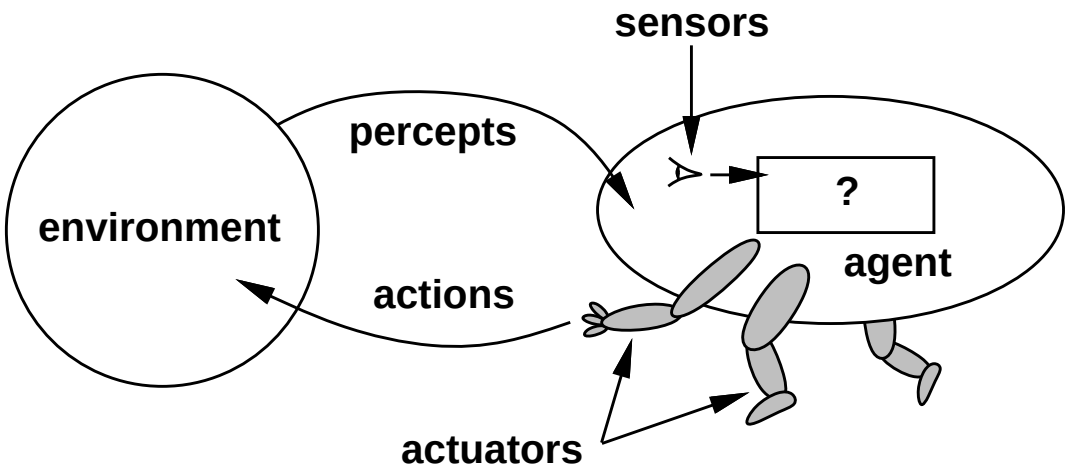
Intelligent Agents

Chapter 2

Outline

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

Agents

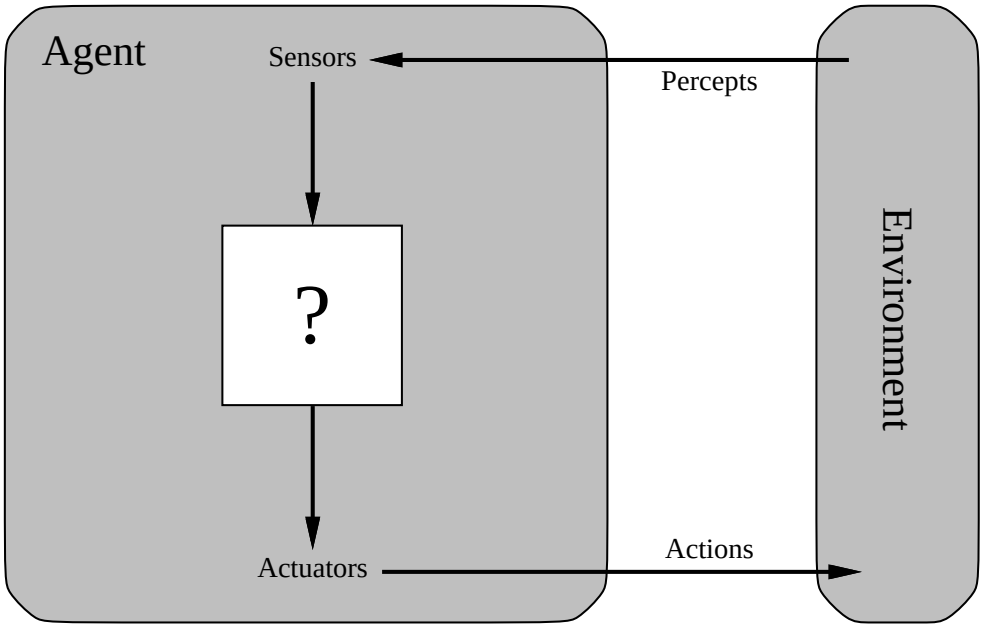


An **agent** is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators

♢ **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

Agents and environments



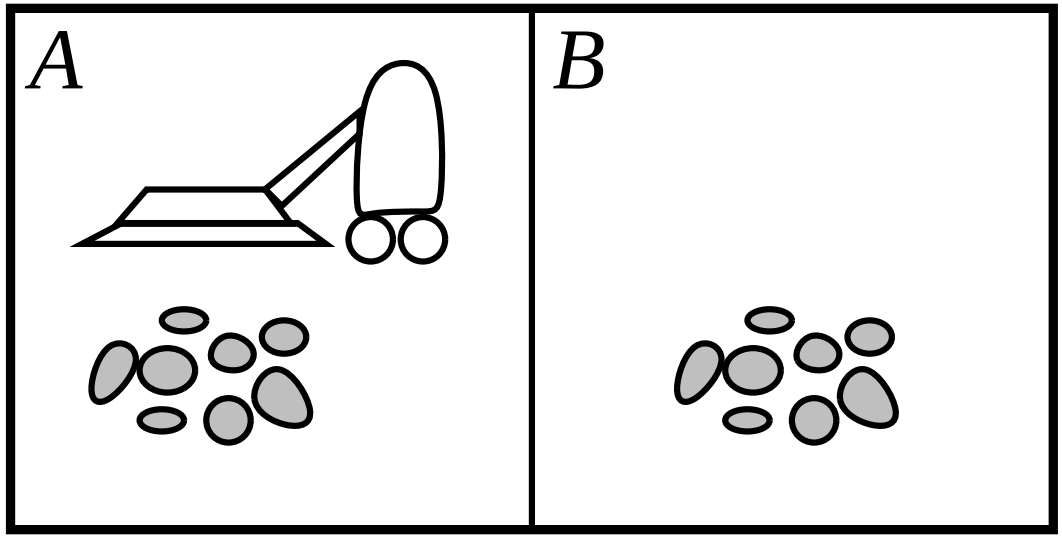
Agents include humans, robots, softbots, thermostats, etc.

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

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
```

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

Rational $\Rightarrow$ exploration, learning, autonomy

Rational agents

"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."

[Russel and Norvig]

Rational agents

Rationality is distinct from omniscience (allknowing with infinite knowledge)

Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering, exploration)

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

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 observable (vs. partially observable)

- An agent's sensors give it access to the complete state of the environment at each point in time.

♢ 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

Environment types (contd.)

♢ 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
- the choice of action in each episode depends only on the episode itself.

♢ Static (vs. dynamic)

- the environment is unchanged while an agent is deliberating.
- the environment is semi-dynamic if the environment itself does not change with the passage of time but the agent's performance score does

Environment types (contd.)

♢ 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

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

Environment types

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

Environment types

	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

	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

	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

	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

The environment type largely determines the agent design

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

Agent functions and programs

An agent is completely specified by the agent function mapping percept sequences to actions

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

The aim is to find a way to implement the rational agent function concisely

Example: Table-lookup agent

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

Drawbacks:

- Huge table
- Takes a long time to build the table
- No autonomy
- Even learning with learning, takes a long time to learn the table entries

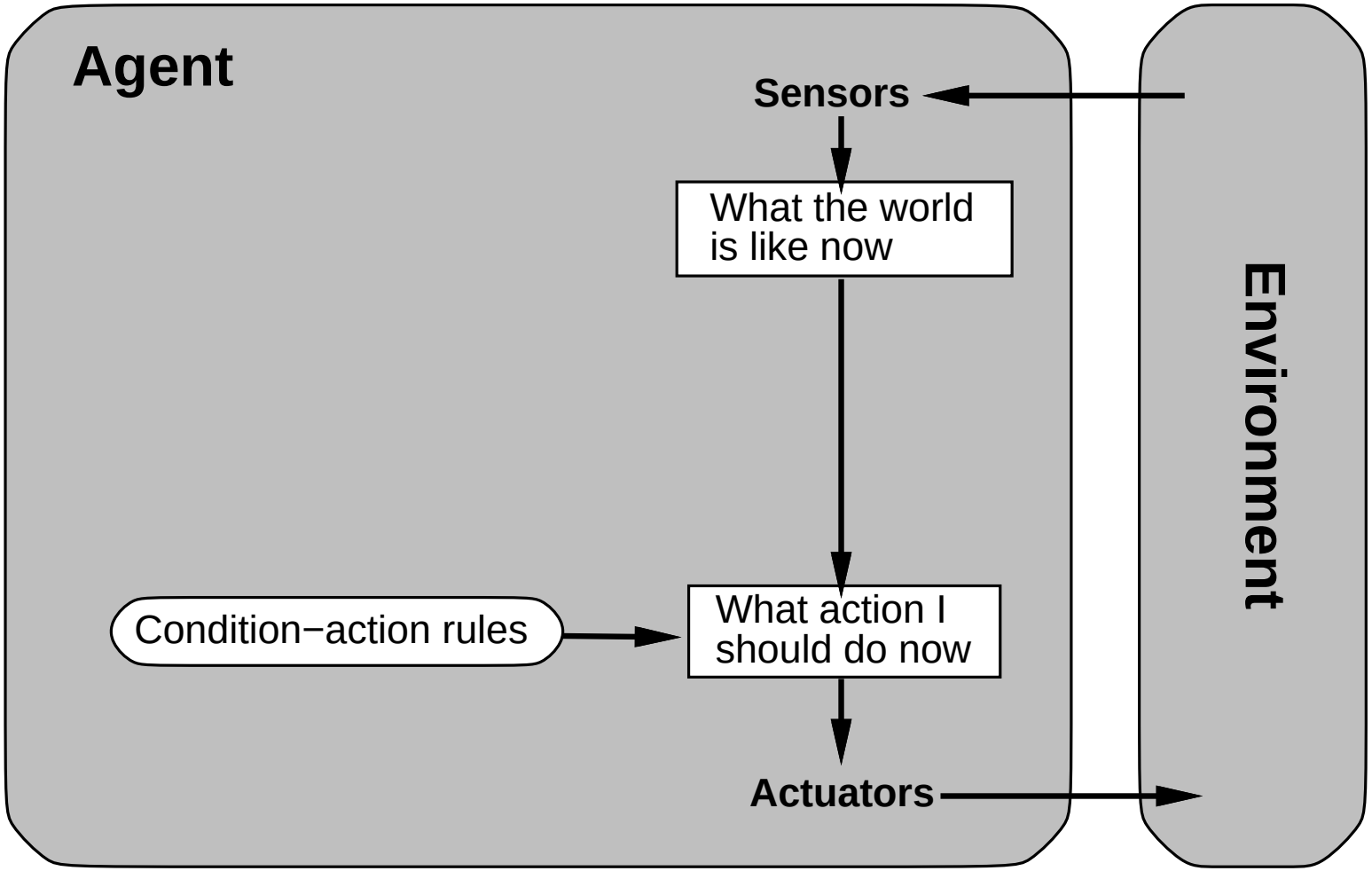
Agent types

Four basic types in order of increasing generality:

- simple reflex agents
- reflex agents with state (Model-based)
- goal-based agents
- utility-based agents

All these can be turned into learning agents

Simple reflex agents



Simple reflex agents

Agent selects actions on the basis of the current percept ignoring percept history

- ♢ Selection based on condition-action rules
- ♢ Randomization

Advantage:

Simplicity, requires only limited resources

Drawback:

It only works if the environment is fully observable

Example

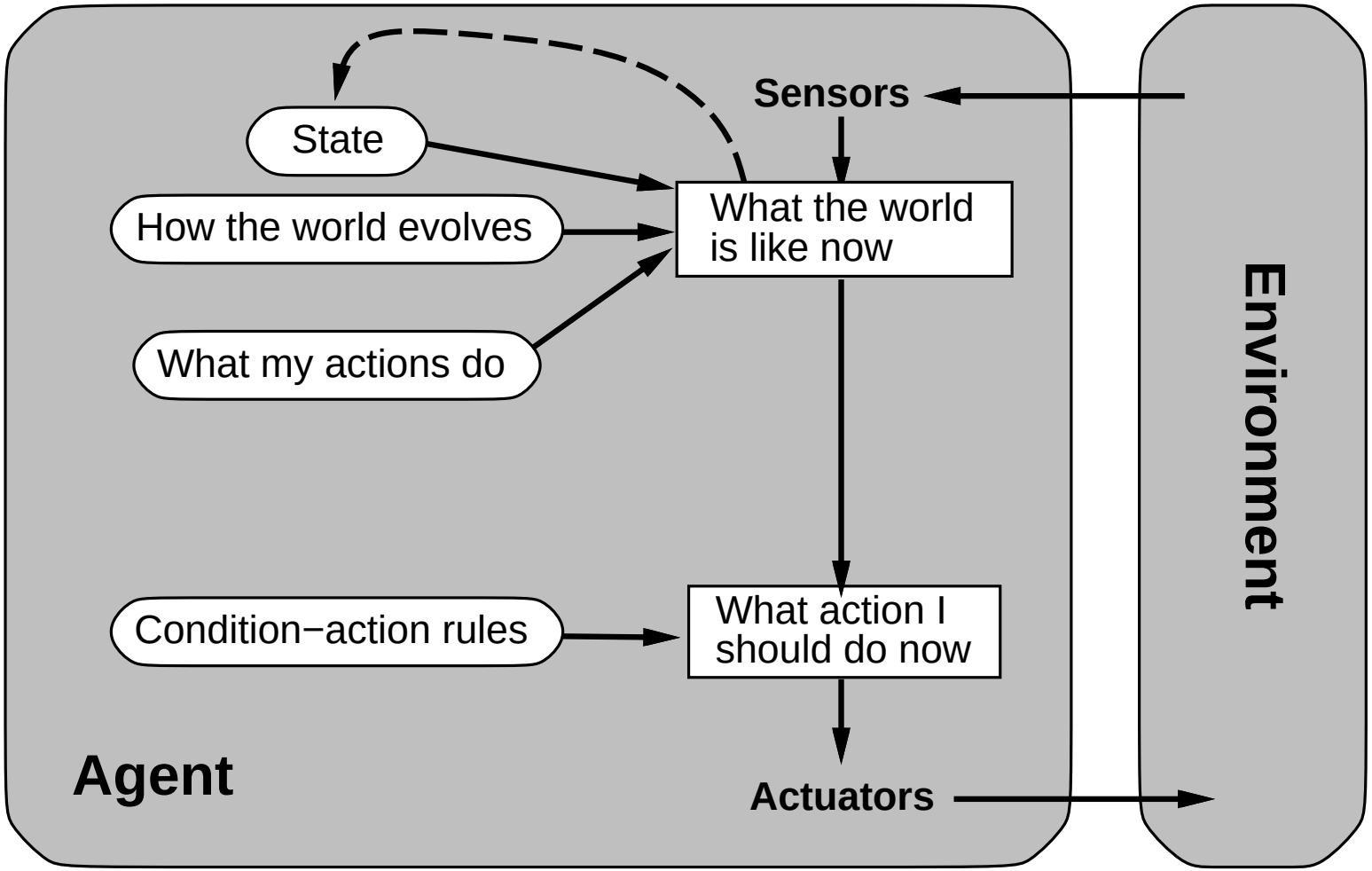
function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

if *status* = *Dirty* **then return** *Suck*
else if *location* = *A* **then return** *Right*
else if *location* = *B* **then return** *Left*

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

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

Reflex agents with state (Model-based)



Reflex agents with state (Model-based)

function REFLEX-AGENT-WITH-STATE(*percept*) **returns** *action*
  static: *state*, a description of the current world state
    rules, a set of condition-action rules

  state ← UPDATE-STATE(*state*, *percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← RULE-ACTION[*rule*]
  state ← UPDATE-STATE(*state*, *action*)
  return *action*

Agent uses model of the world around it to keep track of the parts of the worlds it can not always see

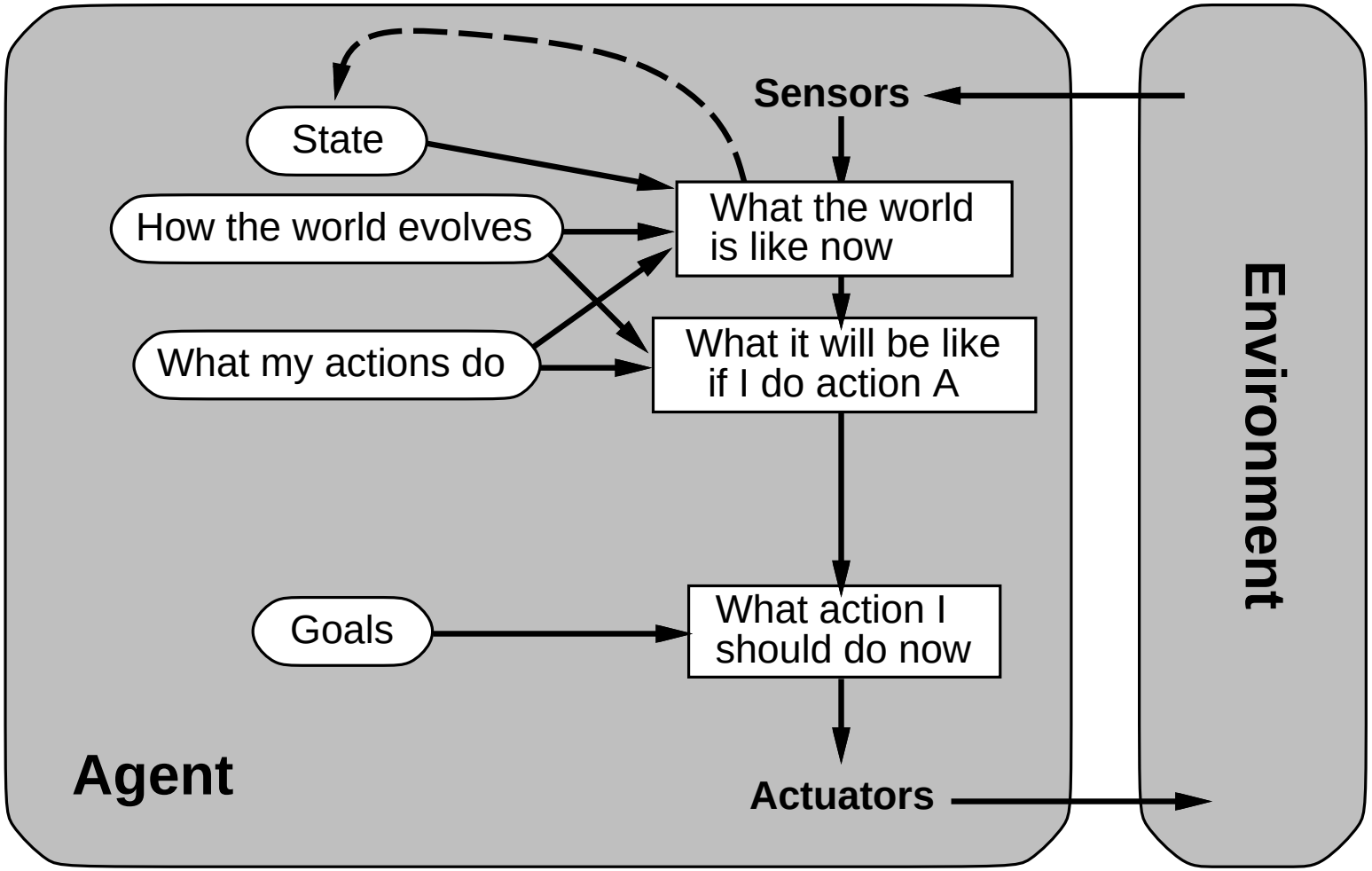
Model:

- ♢ How the world works and evolves independently of the agent
- ♢ How do the actions of the agent affect the world

Example

```
(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)))))))
```

Goal-based agents



Goal-based agents

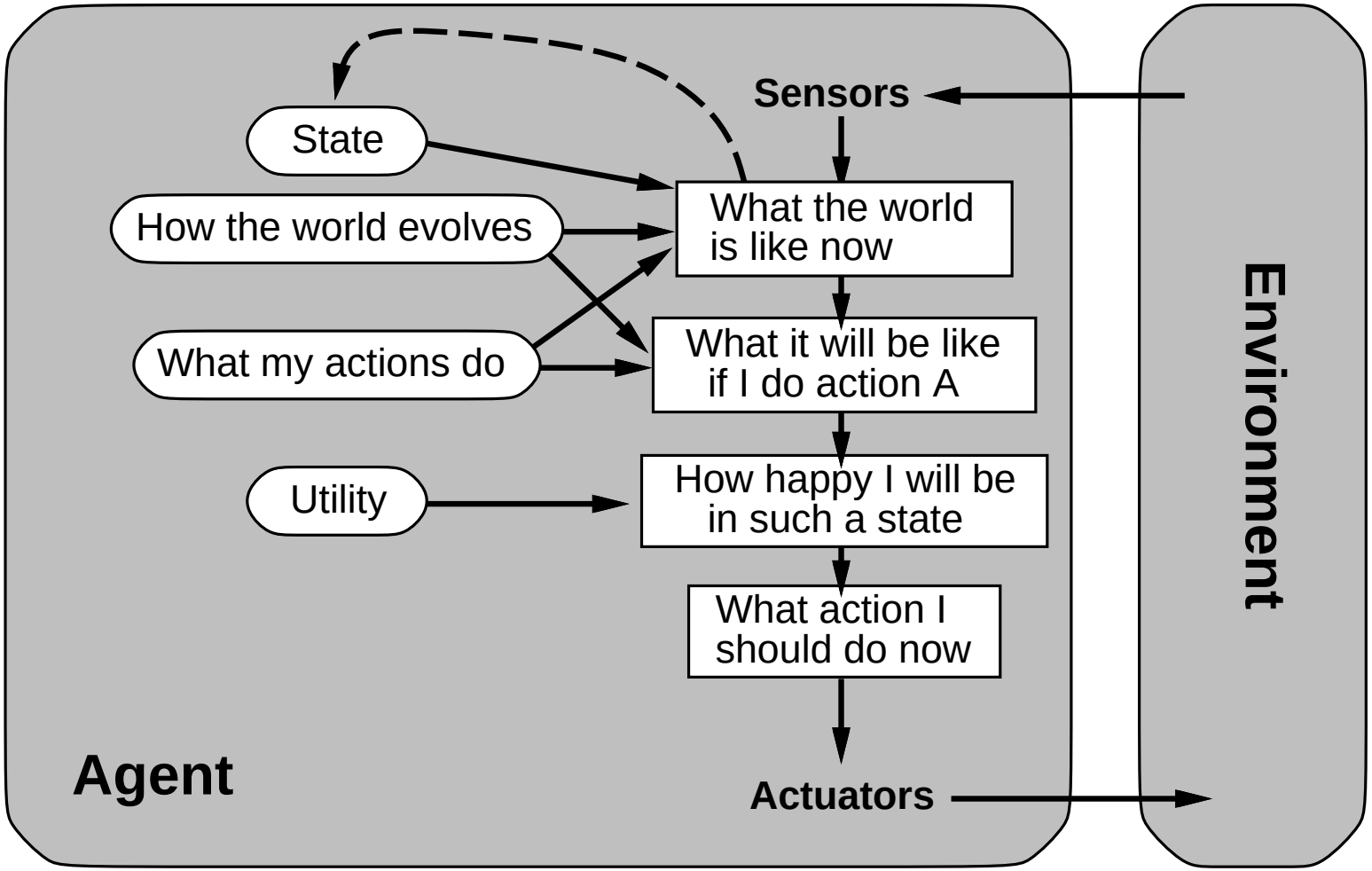
Knowing about the current state of the environment is not always enough to decide what to do

Goal information can ease the action selection process

Goal-based selection can be straightforward or can involve planning

Difference to condition-action rules (simple reflex agents)

Utility-based agents



Utility-based agents

♢ Goal information is not enough to achieve optimal performance

♢ Goals might be contradictory

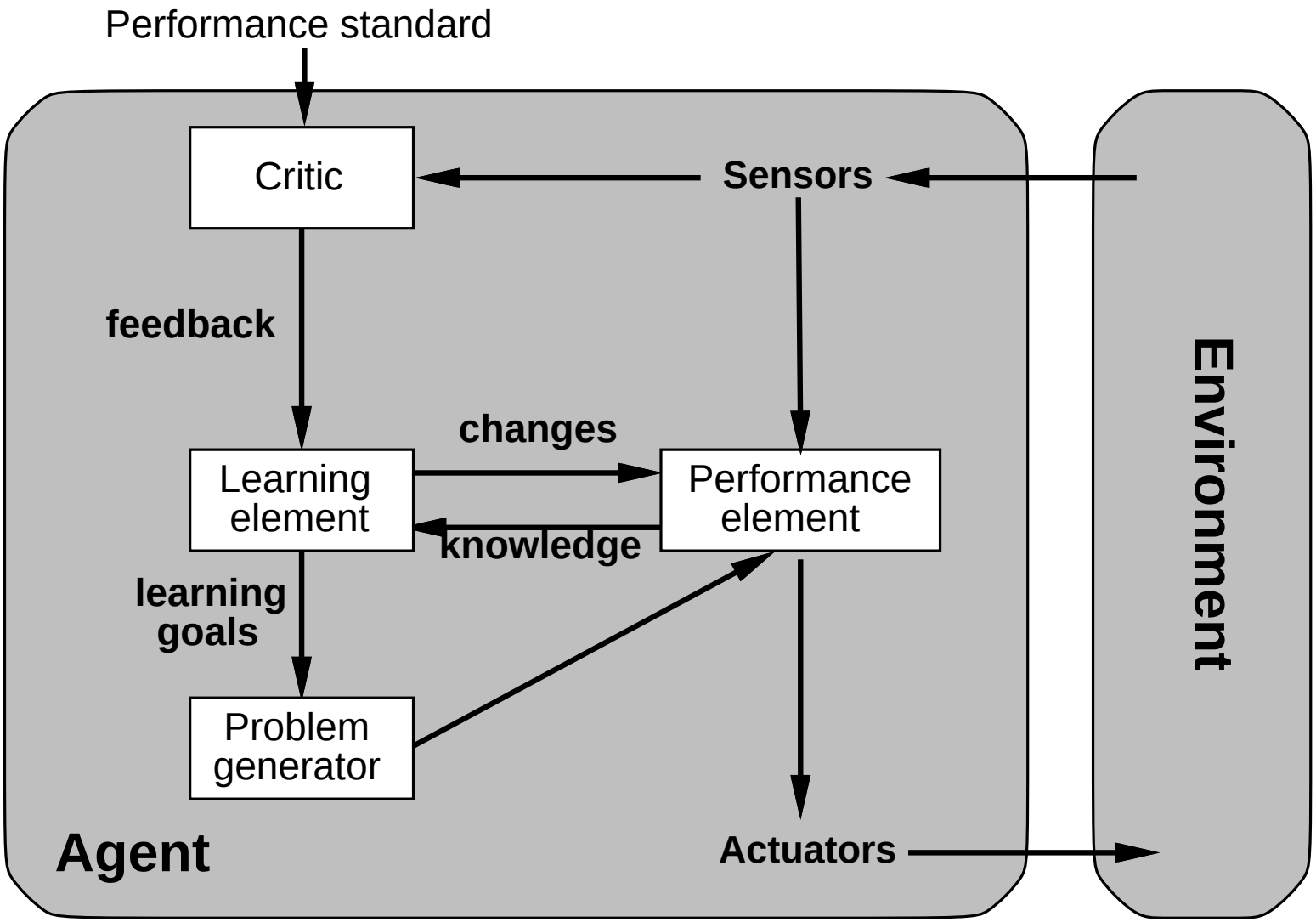
♢ Utility-related considerations can ease the selection of optimal action sequences

♢ Utility function

- Maps state (sequence of states) into a real number
- Resolves contradictions through trade-offs
- Resolves uncertainty through measure for likelihood of success

An agent possessing an explicit utility function can make rational decisions

Learning agents



Learning agents

How to build AI agents

♢ Programming from scratch

♢ Building learning component and teach agent

Learning allows an agent to operate in initially unknown environments and to become more competent than its initial knowledge might allow.

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