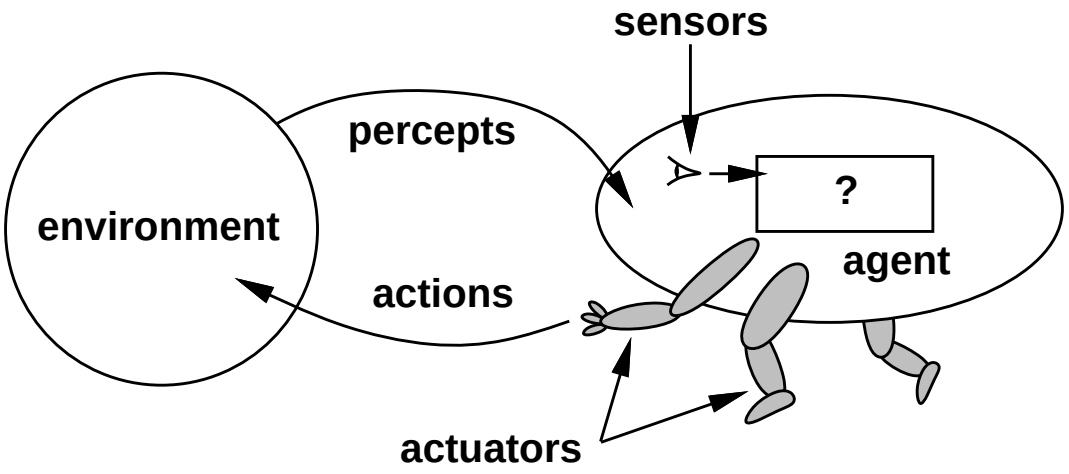
Intelligent Agents

Chapter 2

Outline

- ♢ Agents and environments
- ♢ Rationality
- ♢ PEAS (Performance measure, Environment, Actuators, Sensors)
- ♢ Environment types
- ♢ The structure of agents

Agents and environments



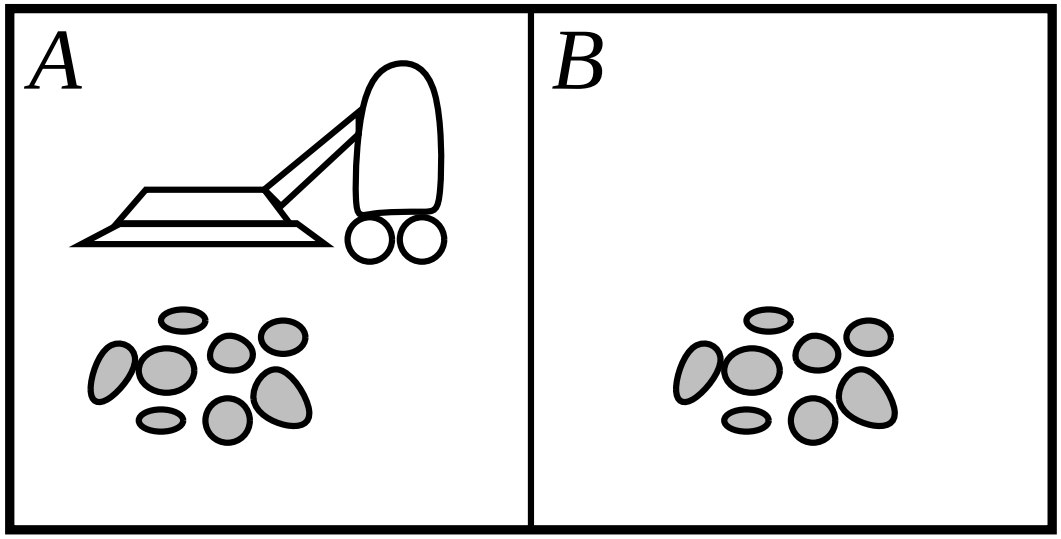
Agents include humans, robots, softbots, thermostats, etc.

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?

Outline

- ♢ Agents and environments
- ♢ **Rationality**
- ♢ PEAS (Performance measure, Environment, Actuators, Sensors)
- ♢ Environment types
- ♢ The structure of agents

Rationality

Fixed performance measure evaluates any sequence of environment states

Rationality

Fixed performance measure evaluates any sequence of environment states

- the amount of dirt cleaned in time T
- one point per each square cleaned up at each time step T?
- one point per clean square per time step, minus one per move?
- penalize for $> k$ dirty squares, electricity, noise?
- having a clean floor

Design performance measures according to what one actually wants in the environment, rather than to how one thinks the agent should behave.

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

Rationality

What is rational at any given time depends on four things:

- The performance measure that defines the criterion of success.
- The agents prior knowledge of the environment.
- The actions that the agent can perform.
- The agents percept sequence to date.

♢ The performance measure awards one point for each clean square at each time step, over a lifetime of 1000 time steps.

♢ The geography of the environment is known a priori but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The Left and Right actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.

♢ The only available actions are Left, Right, and Suck .

♢ The agent correctly perceives its location and whether it contains dirt.

Omniscience, learning, autonomy

Rational $\neq$ omniscient

– percepts may not supply all relevant information

– action outcomes may not be as expected

– assume the agent does not look both ways - is it rational to cross?

– **information gathering** (exploration): doing actions in order to modify future percepts

Hence, rational $\neq$ successful

– (rationality maximises $expected$ performace), given the percept sequence to date.

Autonomy: A rational agent should be learn what it can to compensate for partial or incorrect prior knowledge

–initial knowlege + abilitity to learn

Rational $\Rightarrow$ exploration, learning, autonomy

Outline

♢ Agents and environments

♢ Rationality

♢ **PEAS (Performance measure, Environment, Actuators, Sensors)**

♢ Environment types

♢ The structure of agents

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?? streets/highways, traffic, pedestrians, ,pathholes, puddles, weather, . . .

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

Sensors?? video, speedometer, accelerometer, odometer, engine sensors, 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)

Medical diagnosis system

Performance measure??

Environment??

Actuators??

Sensors??

Medical diagnosis system

Performance measure??: Healthy patient, reduced costs

Environment??: Patient, hospital, staff

Actuators??: Display of questions, tests, diagnoses, treatments

Sensors??: Keyboard entry of symptoms, findings, patient's answers

Interactive English tutor

Performance measure??

Environment??

Actuators??

Sensors??

Interactive English tutor

Performance measure??: Student's score on test

Environment??: Set of students, testing agency

Actuators??: Display of exercises, suggestions, corrections

Sensors??: Keyboard entry

Outline

- ♢ Agents and environments
- ♢ Rationality
- ♢ PEAS (Performance measure, Environment, Actuators, Sensors)
- ♢ **Environment types**
- ♢ The structure of agents

Environment types

♢ Fully observable: sensors give access to the complete state of the environment at each point in time.

– partially obervable: noise, no sensors,

♢ Single/Multi agent Does an agent A (the taxi driver for example) have to treat an object B (another vehicle) as an agent, or can it be treated merely as an object behaving according to the laws of physics?

– whether Bs behavior is best described as maximizing a performance measure whose value depends on agent As behavior.

– competitve/cooperative (chess,taxy,tenis)

♢ Deterministic/stochastic: the next state of the environment is completely determined by the current state and the action of the agent

♢ Uncertain: not fully observable or not deterministic.

♢ Nondeterministic: actions are characterized by their possible outcomes (no probabilities attached as in case of stochastic).

♢ Episodic vs. sequential: the next episode does not depend on the actions in previous episodes.

♢ Static vs. dynamic: environment can change while an agent is deliberating
– semi-dynamic: the environment does not change, but the performance of the agent does (chess with clock)

♢ Discrete vs. continuous: applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent (chess vs. taxi driving)

♢ Known vs. unknown: not to the environment but to agent's knowledge (solitaire vs. video game - know vs. observable)

Environment types

Task Env.	Observable	Agents	Deterministic	Episodic	Static	Discrete

Environment types

Task Env.	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete

Environment types

Task Env.	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete

Environment types

Task Env.	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete

Environment types

Task Env.	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete

Environment types

Task Env.	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with 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	Continous

Environment types

Task Env.	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with 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	Continous
Medical diagnosis	Partially	Multi	Stochastic	Sequential	Dynamic	Continous

Environment types

Task Env.	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with 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	Continous
Medical diagnosis	Partially	Multi	Stochastic	Sequential	Dynamic	Continous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continous

Environment types

Task Env.	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with 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	Continous
Medical diagnosis	Partially	Multi	Stochastic	Sequential	Dynamic	Continous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continous
Part picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continous

Environment types

Task Env.	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with 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	Continous
Medical diagnosis	Partially	Multi	Stochastic	Sequential	Dynamic	Continous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continous
Part picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continous

Environment types

Task Env.	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with 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	Continous
Medical diagnosis	Partially	Multi	Stochastic	Sequential	Dynamic	Continous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continous
Part picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Outline

- ♢ Agents and environments
- ♢ Rationality
- ♢ PEAS (Performance measure, Environment, Actuators, Sensors)
- ♢ Environment types
- ♢ **The structure of agents**

Agent types

agent= architecture + program

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

The table driven agent

function TABLE-DRIVEN-AGENT($percept$) **returns** an action

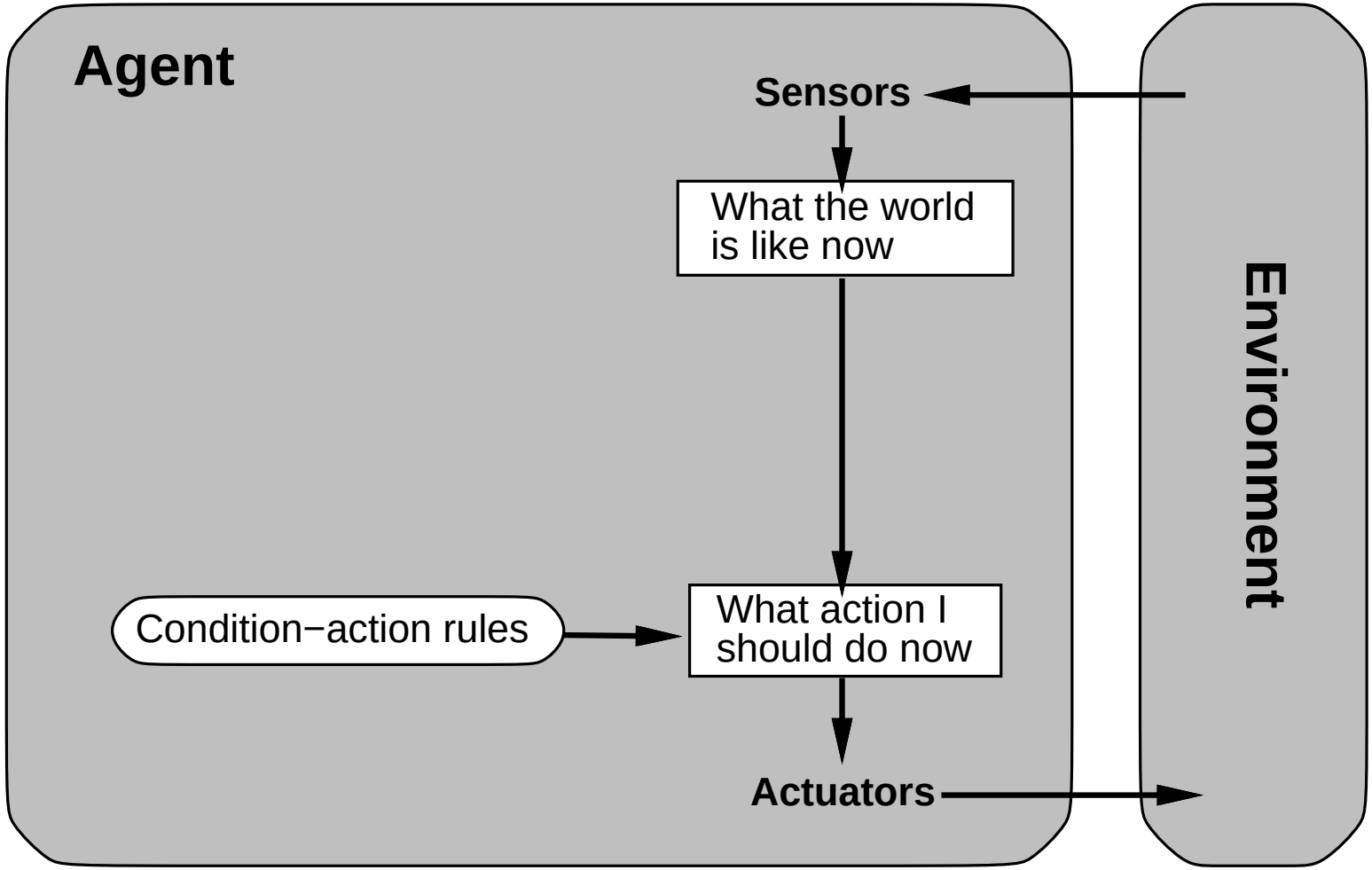
persistent: $percepts$, a sequence, initially empty
$table$, a table of actions, indexed by percept sequences, initially fully specified

append percept to the end of percepts
action $\Leftarrow$ LOOKUP(percepts, table)
return action

Let $\mathcal{P}$ the set of possible percepts $\mathcal{T}$ lifetime of the agent

The lookup table will contain:

Simple reflex agents



Example

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

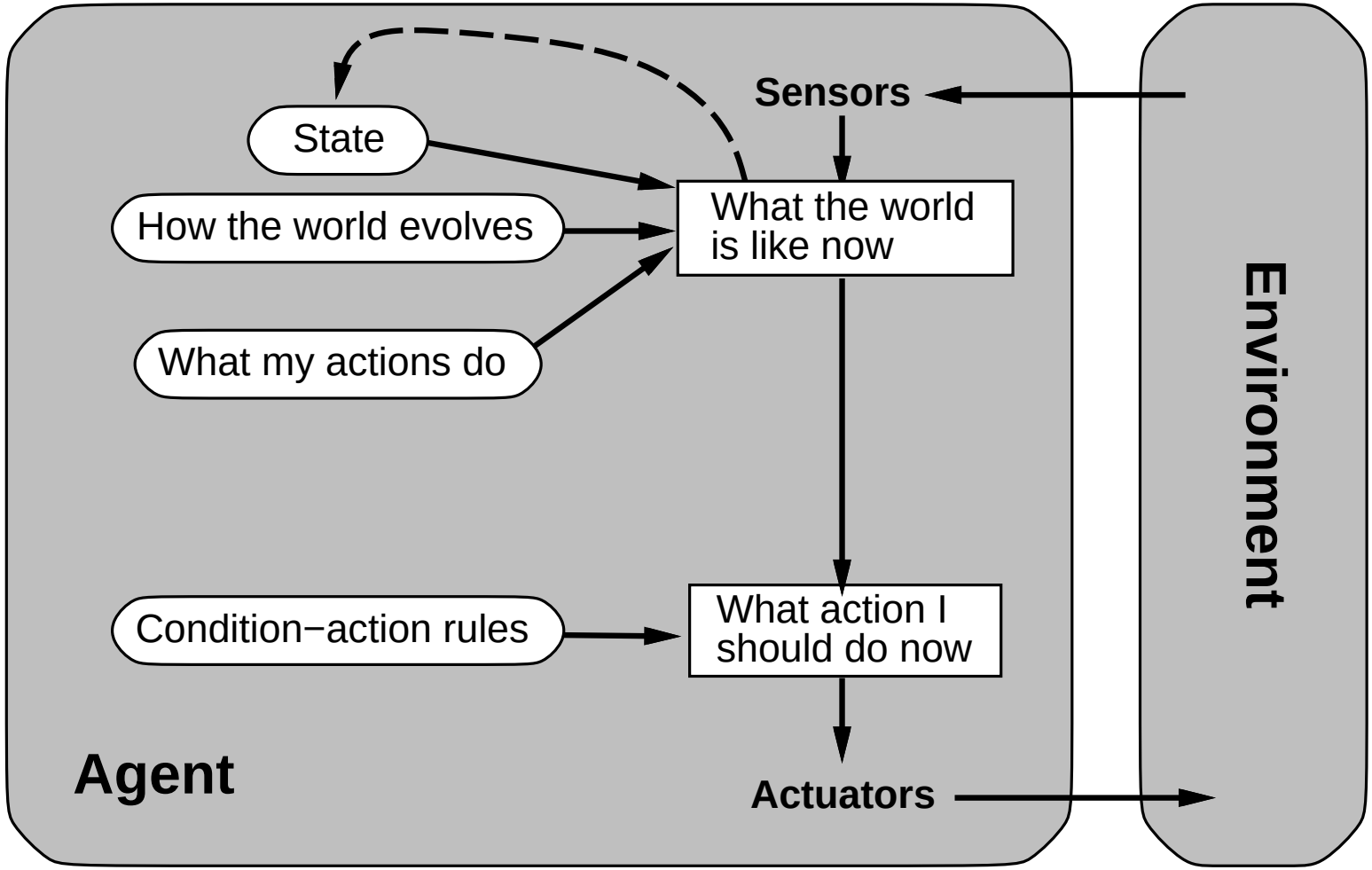
  if *status* = *Dirty* **then return** *Suck*
  else if *location* = *A* **then return** *Right*
  else if *location* = *B* **then return** *Left*

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

Ex: car in front is braking (fully observable, one frame)

Reflex agents with state



Ex: best guess for what the world is like now.

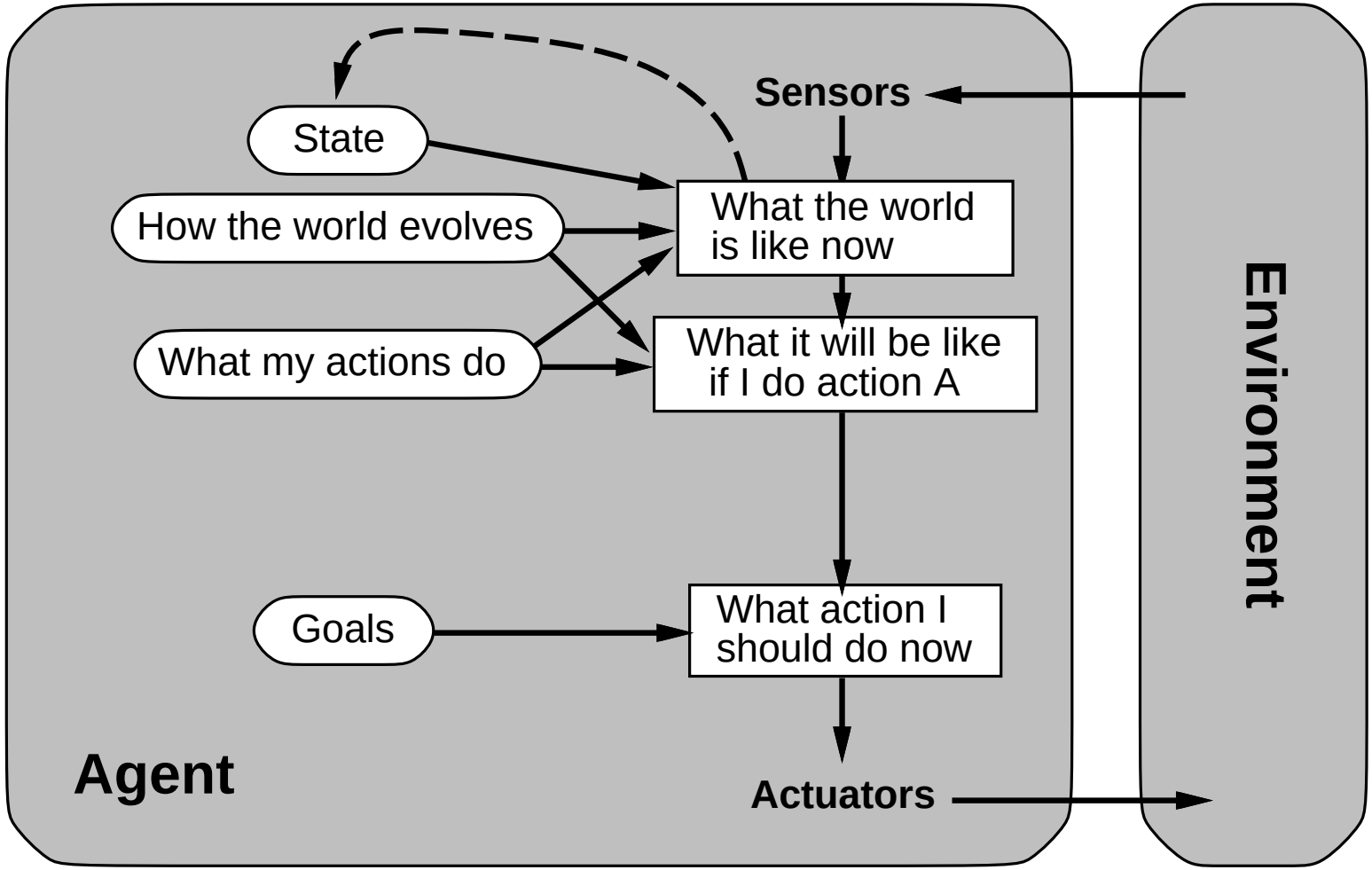
Example

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

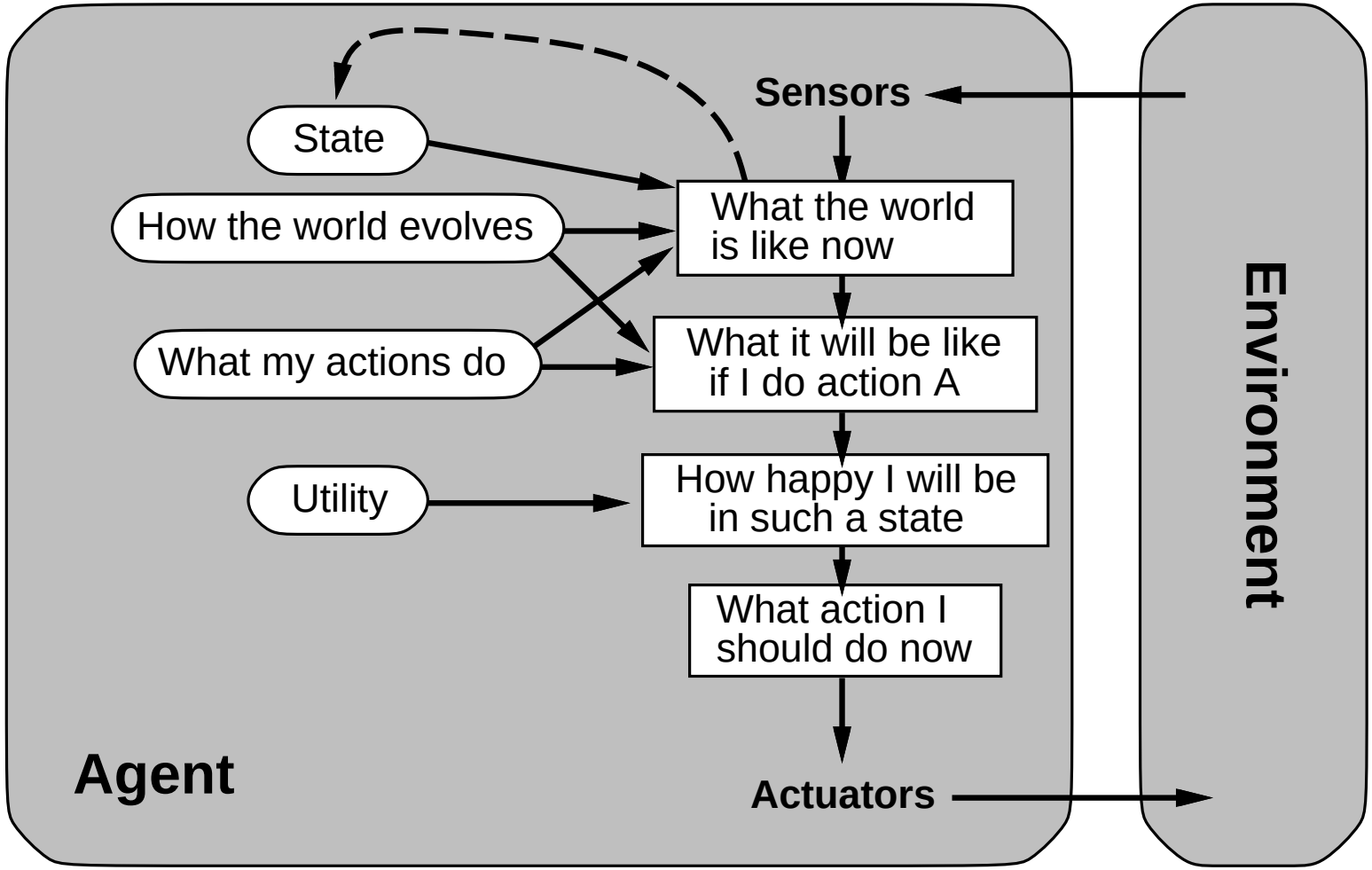
if *status* = *Dirty* **then** . . .

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

Goal-based agents

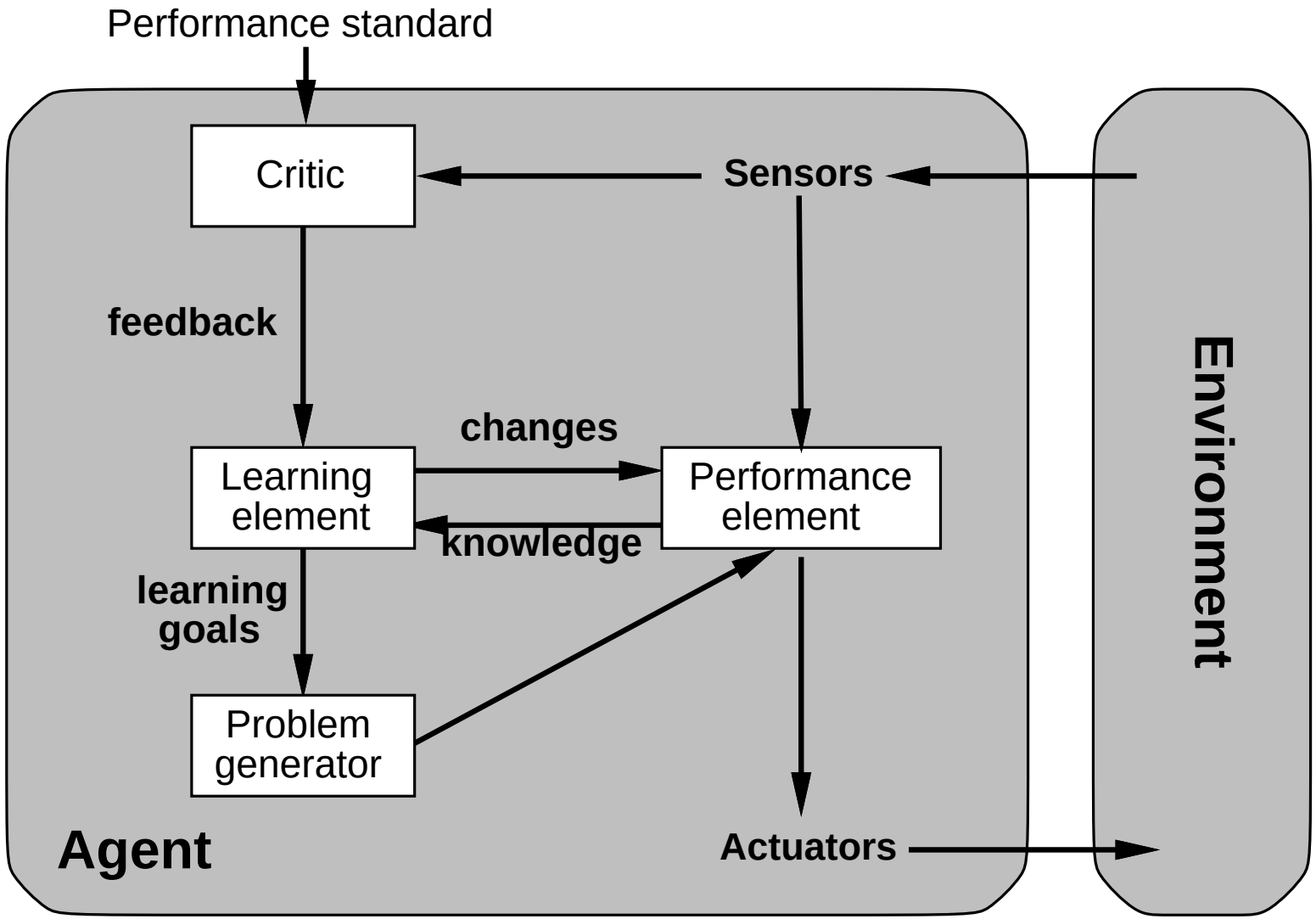


Utility-based agents



Ex: not only happy/unhappy, handle conflicting goals, uncertain goals.

Learning agents



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, learning