Planning and Scheduling: The Agents Metaphor

Prof. Dr.-Ing. Gerhard K. Kraetzschmar

Hochschule Bonn-Rhein-Sieg

b-it Bonn-Aachen International Center for Information Technology

Acknowledgements

- These slides refer to Chapter 2 of the textbook:
  S. Russell and P. Norvig:
  Artificial Intelligence: A Modern Approach
  Prentice Hall, 2003, 2nd Edition (or more recent edition)

- These slides are an adaptation of slides by Min-Yen Kan

- The contributions of these authors are gratefully acknowledged.

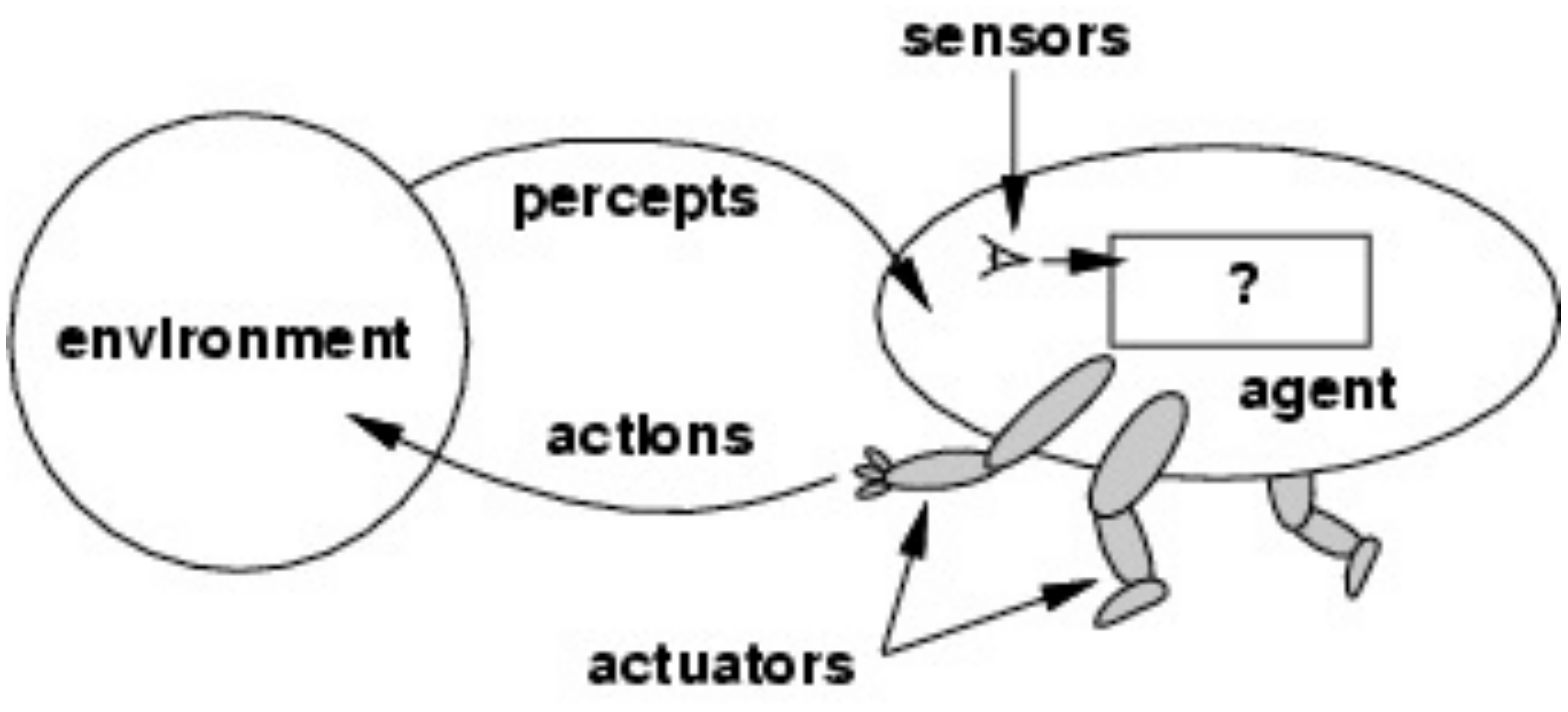
Outline

- 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

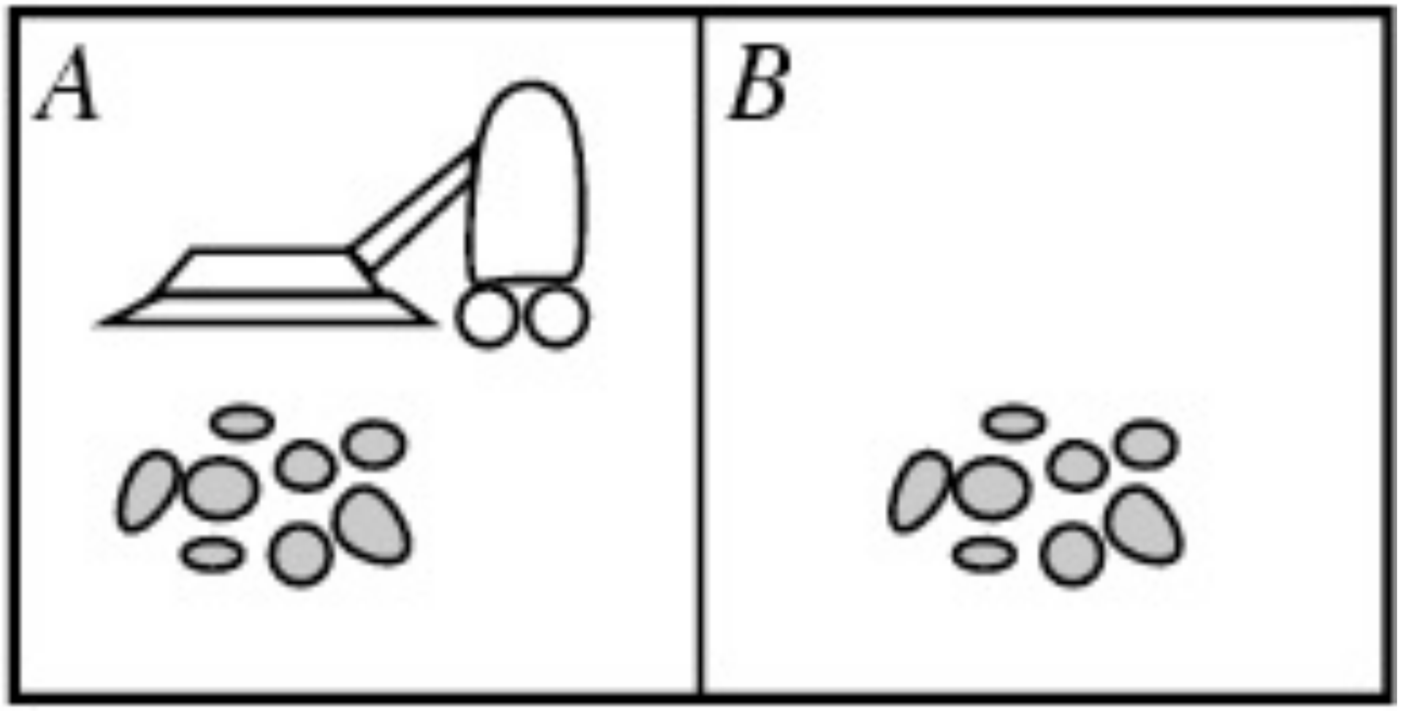


- The agent function maps from percept histories to actions:

$$f : P^* \mapsto A$$

- The agent program runs on the physical architecture to produce f
- agent = architecture + program

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 $Left$
```

Rational Agents

- An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform. The right action will cause the agent to be most successful.
- Performance measure: An objective criterion for success of an agent's behavior.
- Example: the performance measure of a vacuum-cleaner agent could be
  - the amount of dirt cleaned up,
  - the amount of time taken,
  - the amount of electricity consumed,
  - the amount of noise generated, etc.

Rational Agents

- Rational Agent: 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
- Rationality is distinct from omniscience (all-knowing 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 (with ability to learn and adapt)

PEAS

- We must first specify the setting for intelligent agent design
- This is done in terms of PEAS descriptions:
  - Performance measure
  - Environment
  - Actuators
  - Sensors

- Consider, e.g., the task of designing an automated taxi driver:
- Agent: taxi driver
  - Performance measure: safe, fast, legal, comfortable trip, maximize profits
  - Environment: roads, other traffic, pedestrians, customers
  - Actuators: steering wheel, accelerator, brake, signal, horn
  - Sensors: cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard

PEAS

- Agent: medical diagnosis system
  - Performance measure: healthy patient, minimize costs, lawsuits
  - Environment: patient, hospital, staff
  - Actuators: screen display (questions, tests, diagnoses, treatments, referrals)
  - Sensors: keyboard (entry of symptoms, findings, patient's answers)
- Agent: part-picking robot
  - Performance measure: percentage of parts in correct bins
  - Environment: conveyor belt with parts, bins
  - Actuators: jointed arm and hand
  - Sensors: camera, joint angle sensors
- Agent: interactive English tutor
  - Performance measure: maximize student's score on test
  - Environment: set of students
  - Actuators: screen display (exercises, suggestions, corrections)
  - Sensors: keyboard

Environment Types

- Fully observable vs. partially observable:
  - Agent has 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.)
- 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), and the choice of action in each episode depends only on the episode itself.

Environment Types

- Static (vs. dynamic):
  - The environment is unchanged while an agent is deliberating. (The environment is semidynamic if the environment itself does not change with the passage of time but the agent's performance score does)
- 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

	Chess w. clock	Chess w.o. clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, and multiagent

Agent Functions and Programs

- An agent is completely specified by the agent function mapping percept sequences to actions
- One agent function (or a small equivalence class) is rational
- Aim: find a way to implement the rational agent function concisely

Table-Lookup Agent

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

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

Agent Program For a Vacuum-Cleaner Agent

```
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 $Left$
```

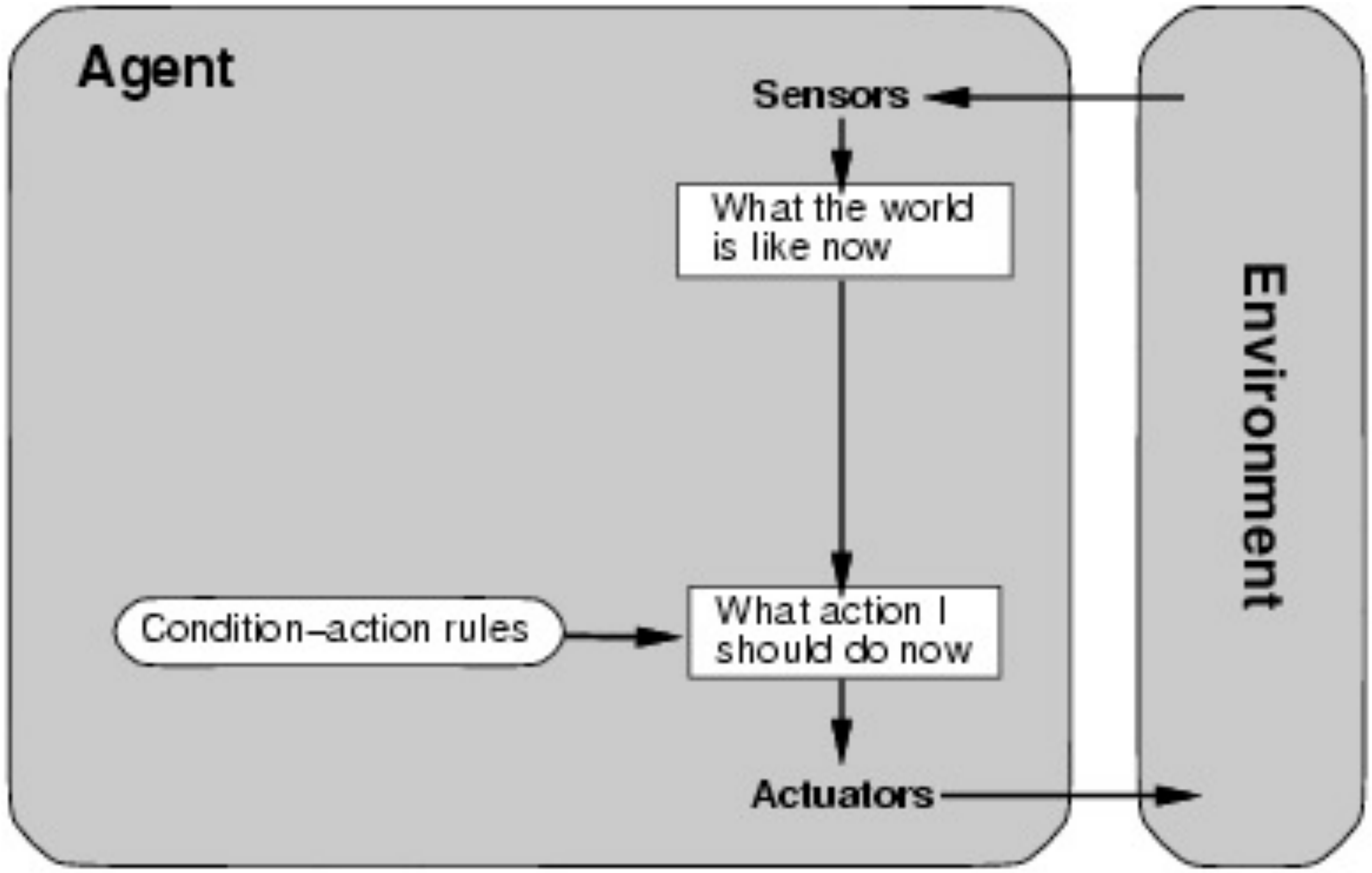
```
(setq joe (make-agent :name 'joe :body (make-agent-body)
                      :program (make-reflex-vacuum-agent-program)))

(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)))))
```

Agent Types

- Four basic types in order of increasing generality:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

Simple Reflex Agents



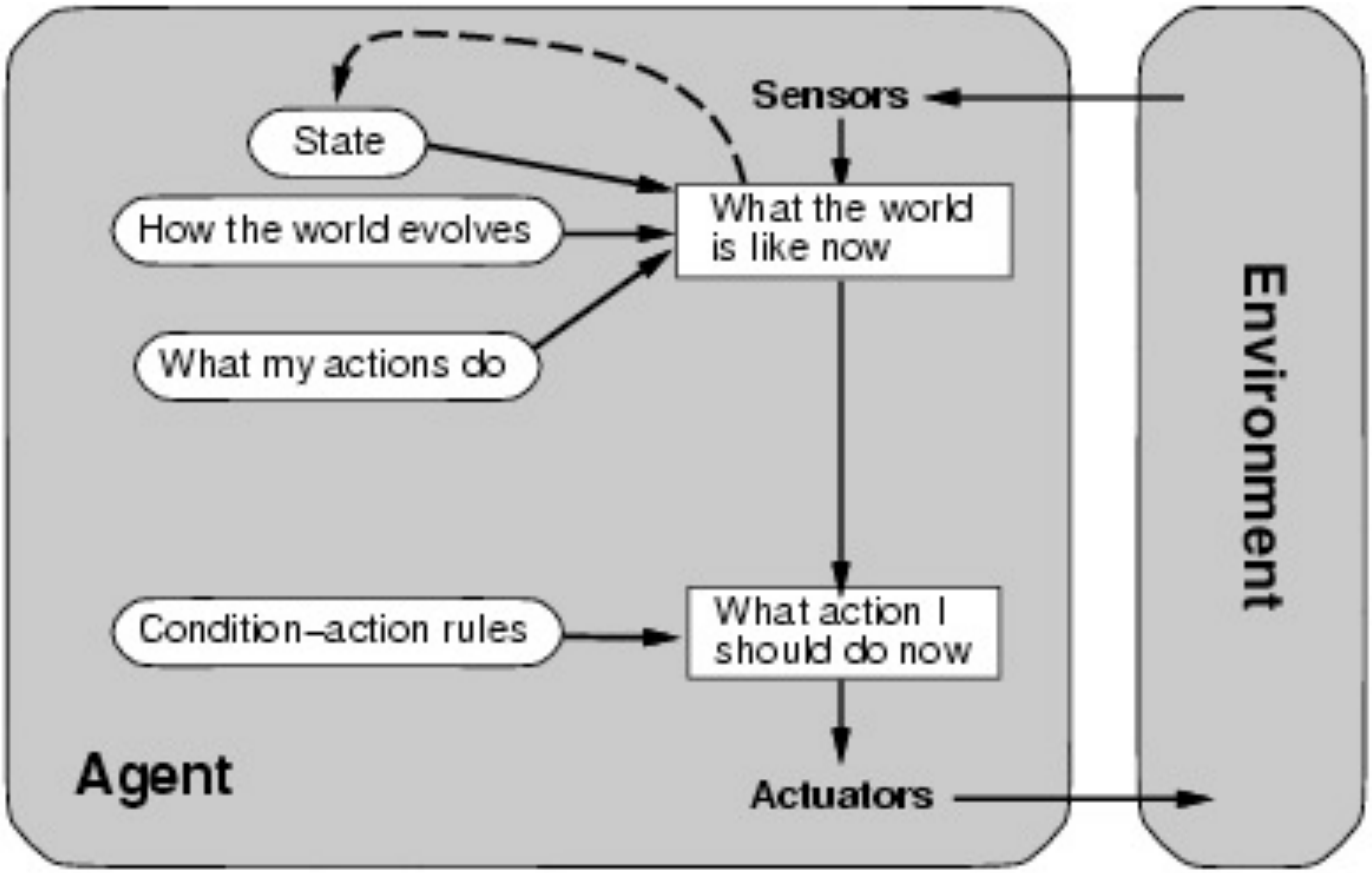
Simple Reflex Agents

```
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))))))
```

Model-Based Reflex Agents



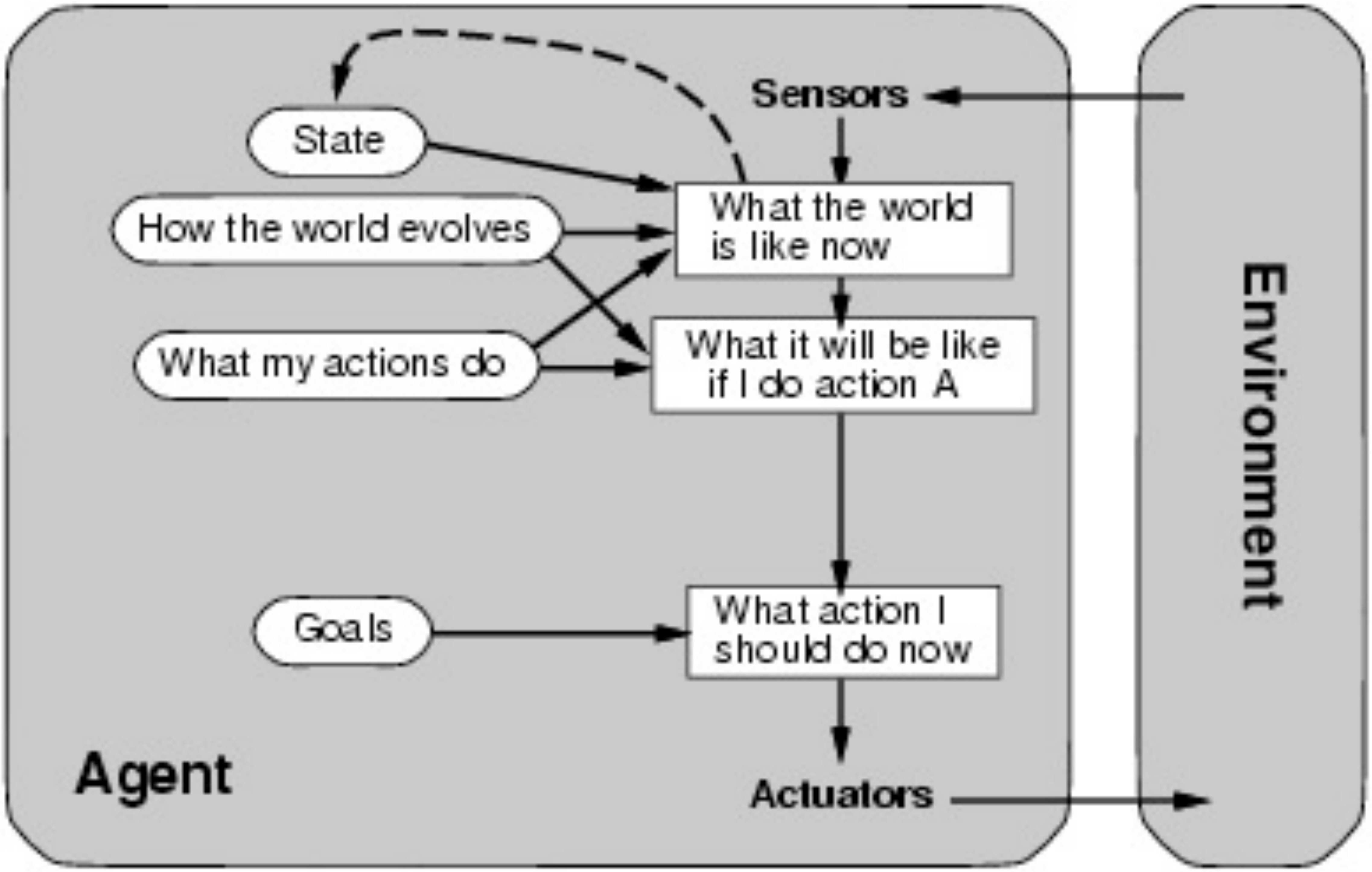
Model-Based Reflex Agents

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

   if status = Dirty then . . .
```

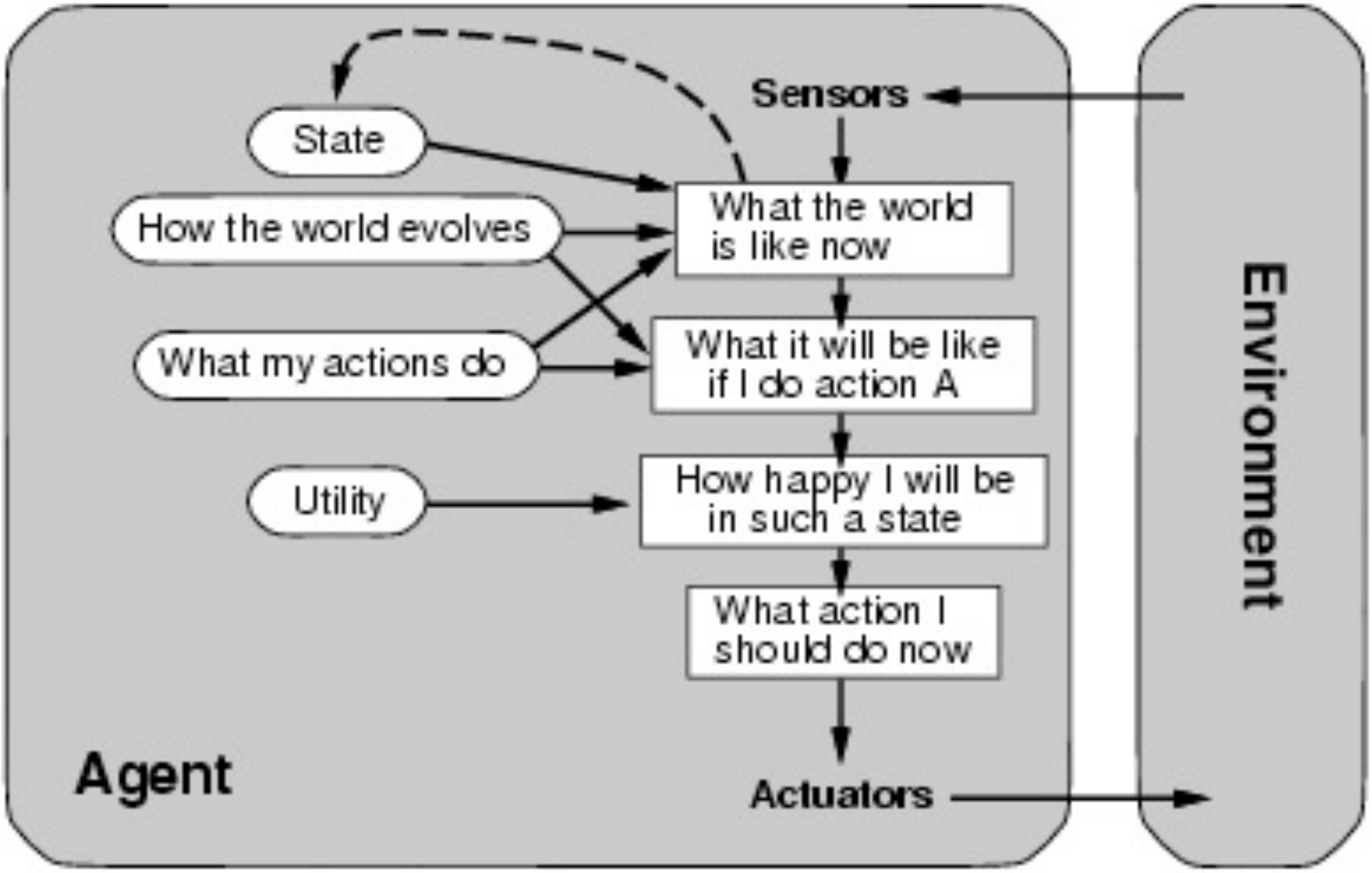
```
(defun make-reflex-vacuum-agent-with-state-program ()
  (let ((last-A infinity) (last-B infinity))
  #'(lambda (percept)
      (let ((location (first percept)) (status (second percept)))
        (incf last-A) (incf last-B)
        (cond
         ((eq status 'dirty)
          (if (eq location 'A) (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



Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

Utility-Based Agents

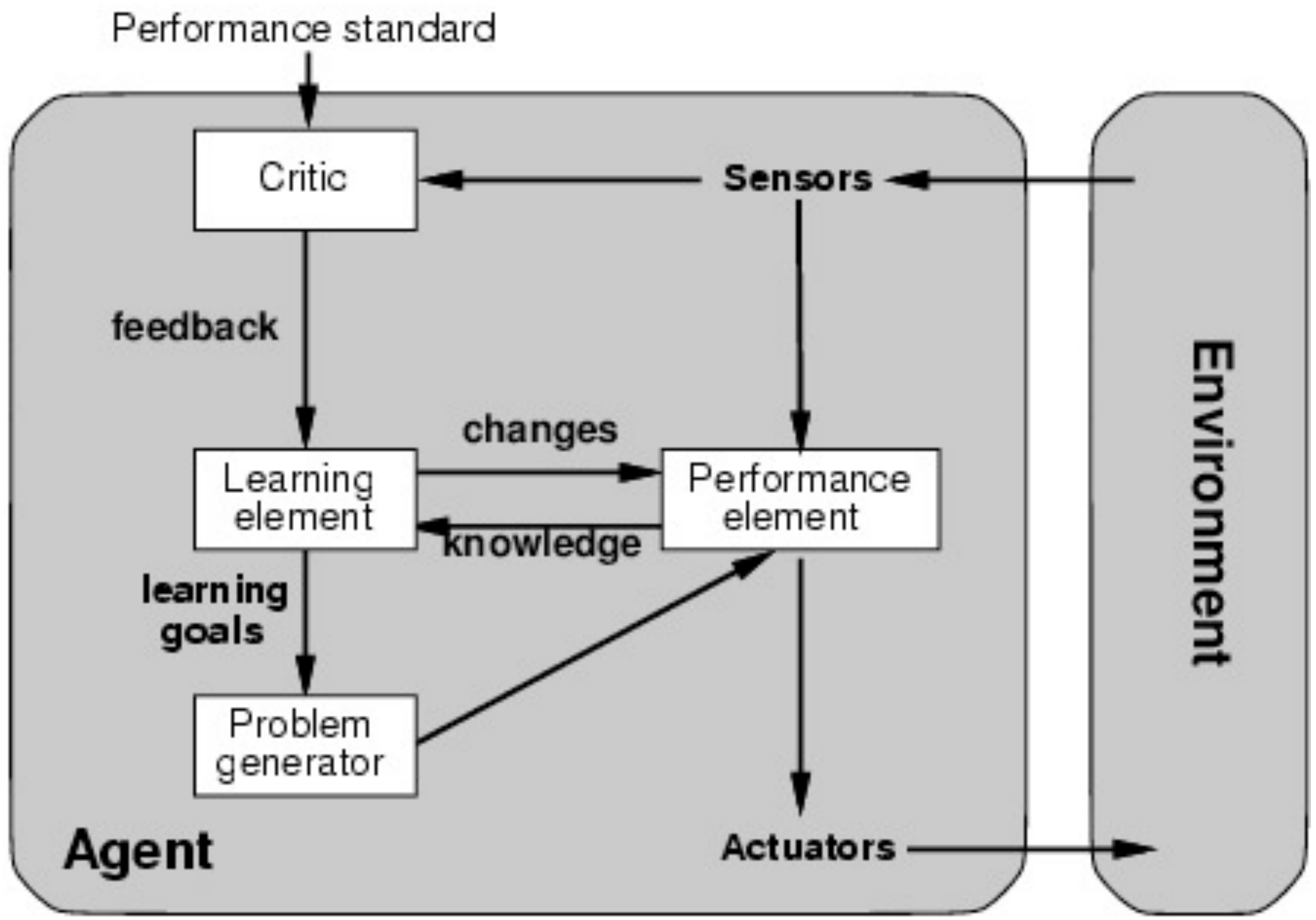


Sensors
State
How the world evolves
What my actions do
Utility
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Actuators
Agent
Environment

Advanced Agent Architectures

- These basic agent types still remain simple and usually cannot deal with more complex environments (dynamic, partially-observable, etc.)
- How do we account e.g. for learning?
  - (see next slide)
  - (see project XPERO)
- Interesting architectures of agents have been investigated in robotics
  - 3T architecture
  - Dynamo, WayFinder
- The architecture of complex agents is still an open research issue!
  - (see projects like DESIRE, ROSTA, XPERO, BRICS)

Learning Agents


Performance standard
Critic
Sensors
feedback
changes
Learning element
Performance element
knowledge
learning goals
Problem generator
Agent
Actuators
Environment