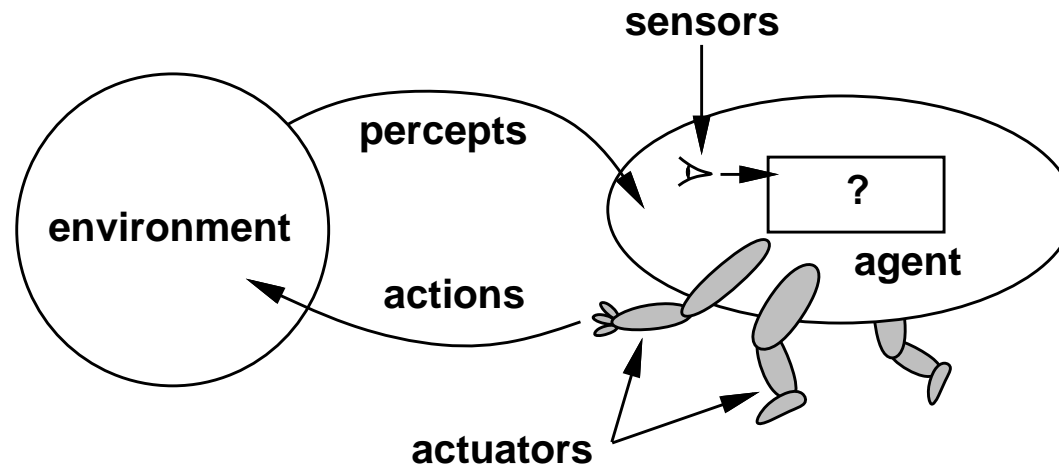


Last update: September 2, 2008

# INTELLIGENT AGENTS

## CMSC 421: CHAPTER 2

# Agents and environments



Russell & Norvig's book is organized around the concept of **intelligent agents**  
◇ 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  $f$

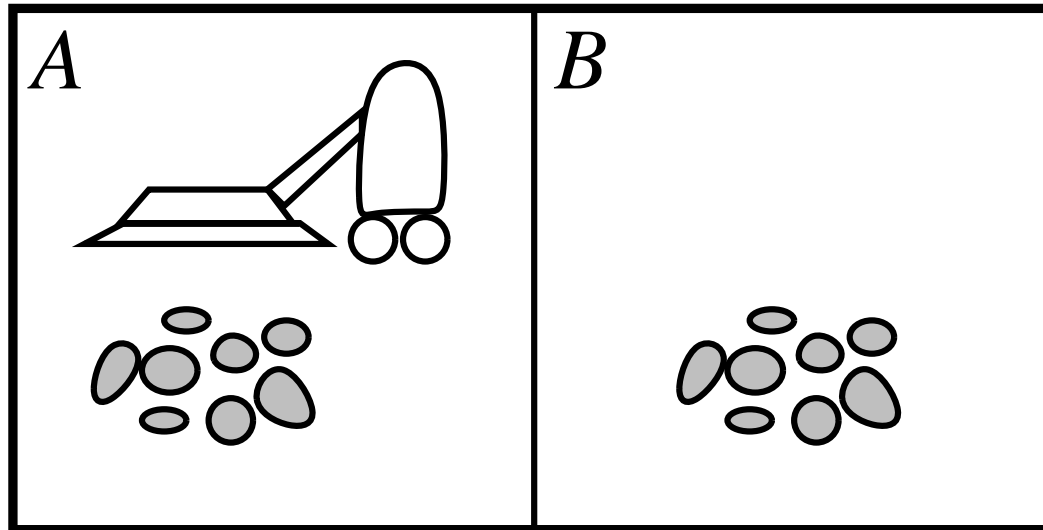
# Chapter 2 - Purpose and Outline

Purpose of Chapter 2:

basic concepts relating to agents

- ◇ Agents and environments
- ◇ Rationality
- ◇ The PEAS model of an environment
- ◇ Environment types
- ◇ Agent types

## Vacuum-cleaner world



Percepts: location and contents, e.g.,  $[A, \textit{Dirty}]$

Actions: *Left*, *Right*, *Suck*, *NoOp*

## A vacuum-cleaner agent

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

```
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

Rational  $\Rightarrow$  exploration, learning, autonomy

# 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

	Solitaire	Backgammon	Internet shopping	Taxi
<u>Fully observable?</u>				
<u>Deterministic?</u>				
<u>Episodic?</u>				
<u>Static?</u>				
<u>Discrete?</u>				
<u>Single-agent?</u>				

## Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
<u>Fully observable?</u>	No	Yes	No	No
<u>Deterministic?</u>				
<u>Episodic?</u>				
<u>Static?</u>				
<u>Discrete?</u>				
<u>Single-agent?</u>				

## Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
<u>Fully observable?</u>	No	Yes	No	No
<u>Deterministic?</u>	Yes*	No	Partly	No
<u>Episodic?</u>				
<u>Static?</u>				
<u>Discrete?</u>				
<u>Single-agent?</u>				

\*After the cards have been dealt

## Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
<u>Fully observable?</u>	No	Yes	No	No
<u>Deterministic?</u>	Yes*	No	Partly	No
<u>Episodic?</u>	No	No	No	No
<u>Static?</u>				
<u>Discrete?</u>				
<u>Single-agent?</u>				

\*After the cards have been dealt

## Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
<u>Fully observable?</u>	No	Yes	No	No
<u>Deterministic?</u>	Yes*	No	Partly	No
<u>Episodic?</u>	No	No	No	No
<u>Static?</u>	Yes	Semi	Semi	No
<u>Discrete?</u>				
<u>Single-agent?</u>				

\*After the cards have been dealt

## Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
<u>Fully observable?</u>	No	Yes	No	No
<u>Deterministic?</u>	Yes*	No	Partly	No
<u>Episodic?</u>	No	No	No	No
<u>Static?</u>	Yes	Semi	Semi	No
<u>Discrete?</u>	Yes	Yes	Yes	No
<u>Single-agent?</u>				

\*After the cards have been dealt

## Environment types

	Solitaire	Backgammon	Internet shopping	Taxi
<u>Fully observable?</u>	No	Yes	No	No
<u>Deterministic?</u>	Yes*	No	Partly	No
<u>Episodic?</u>	No	No	No	No
<u>Static?</u>	Yes	Semi	Semi	No
<u>Discrete?</u>	Yes	Yes	Yes	No
<u>Single-agent?</u>	Yes	No	No	No

\*After the cards have been dealt

### The environment type largely determines the agent design

Real world: partially observable, stochastic, sequential, dynamic, continuous, multi-agent

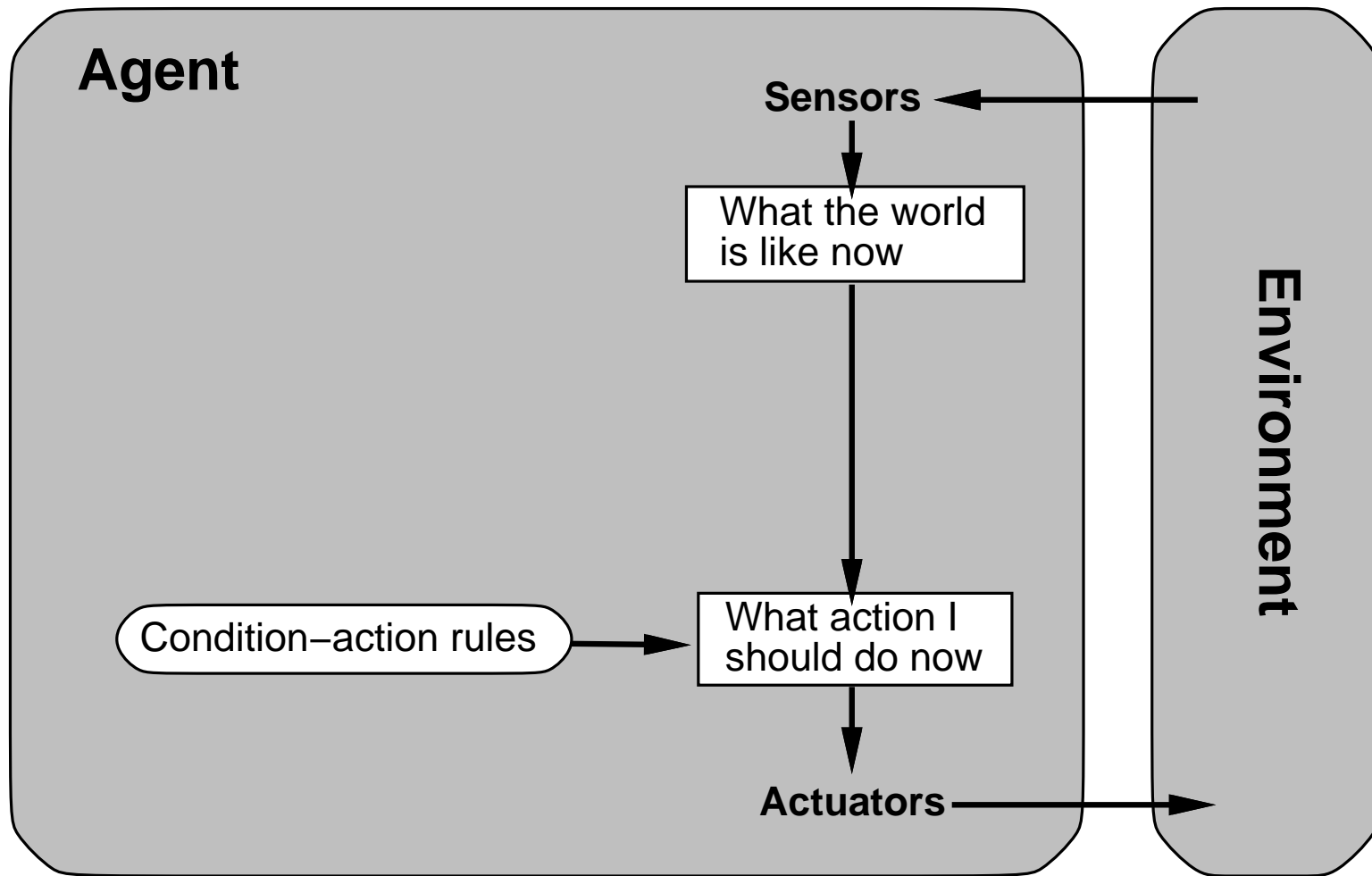
# Agent types

Four basic types in order of increasing generality:

- simple reflex agents
- reflex agents with state
- goal-based agents
- utility-based agents

All of these can be turned into learning agents

# Simple reflex agents



## Example

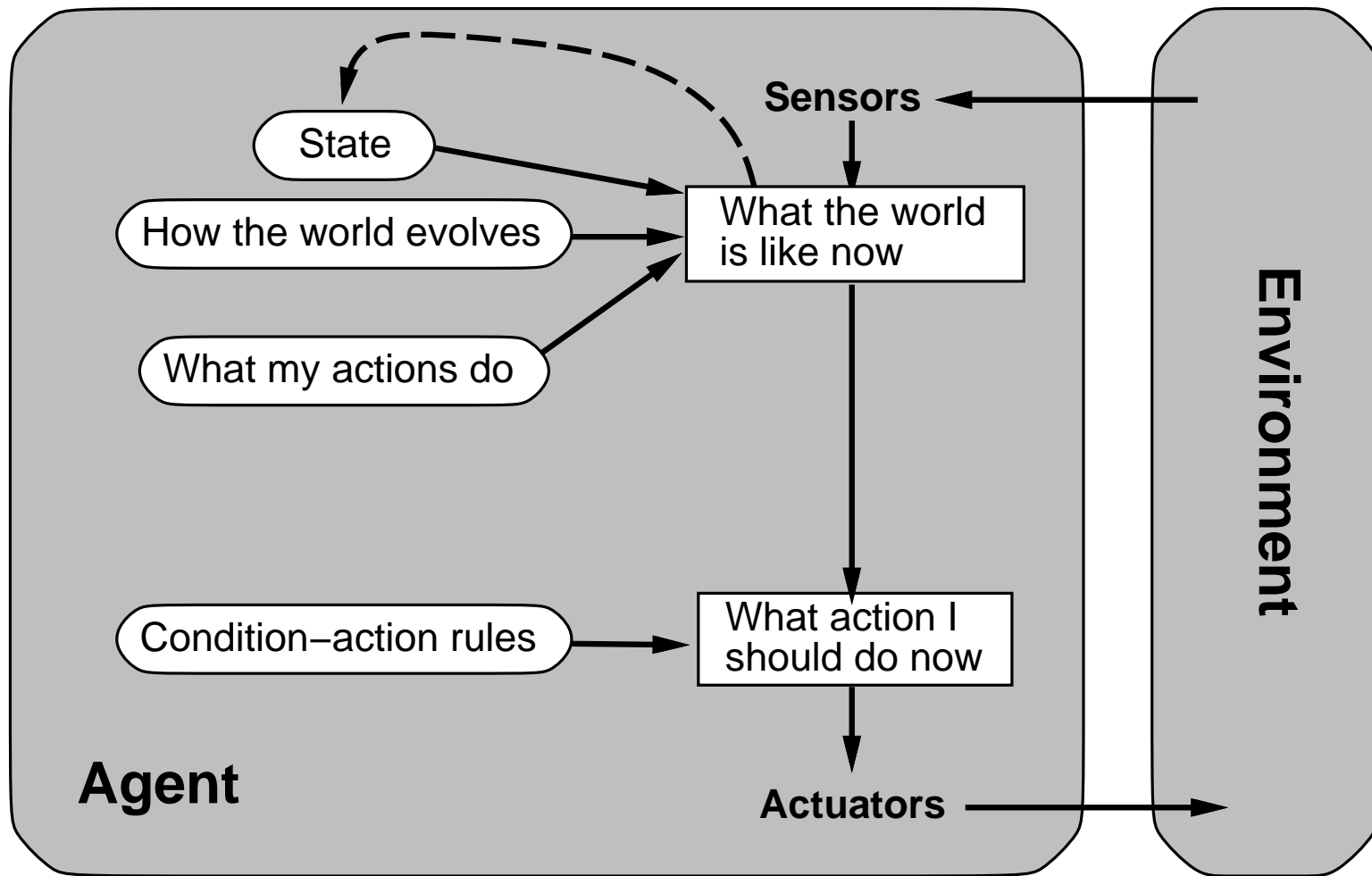
**function** REFLEX-VACUUM-AGENT( [*location, status*]) **returns** an action

**if** *status* = *Dirty* **then return** *Suck*

**else if** *location* = *A* **then return** *Right*

**else if** *location* = *B* **then return** *Left*

# Reflex agents with state



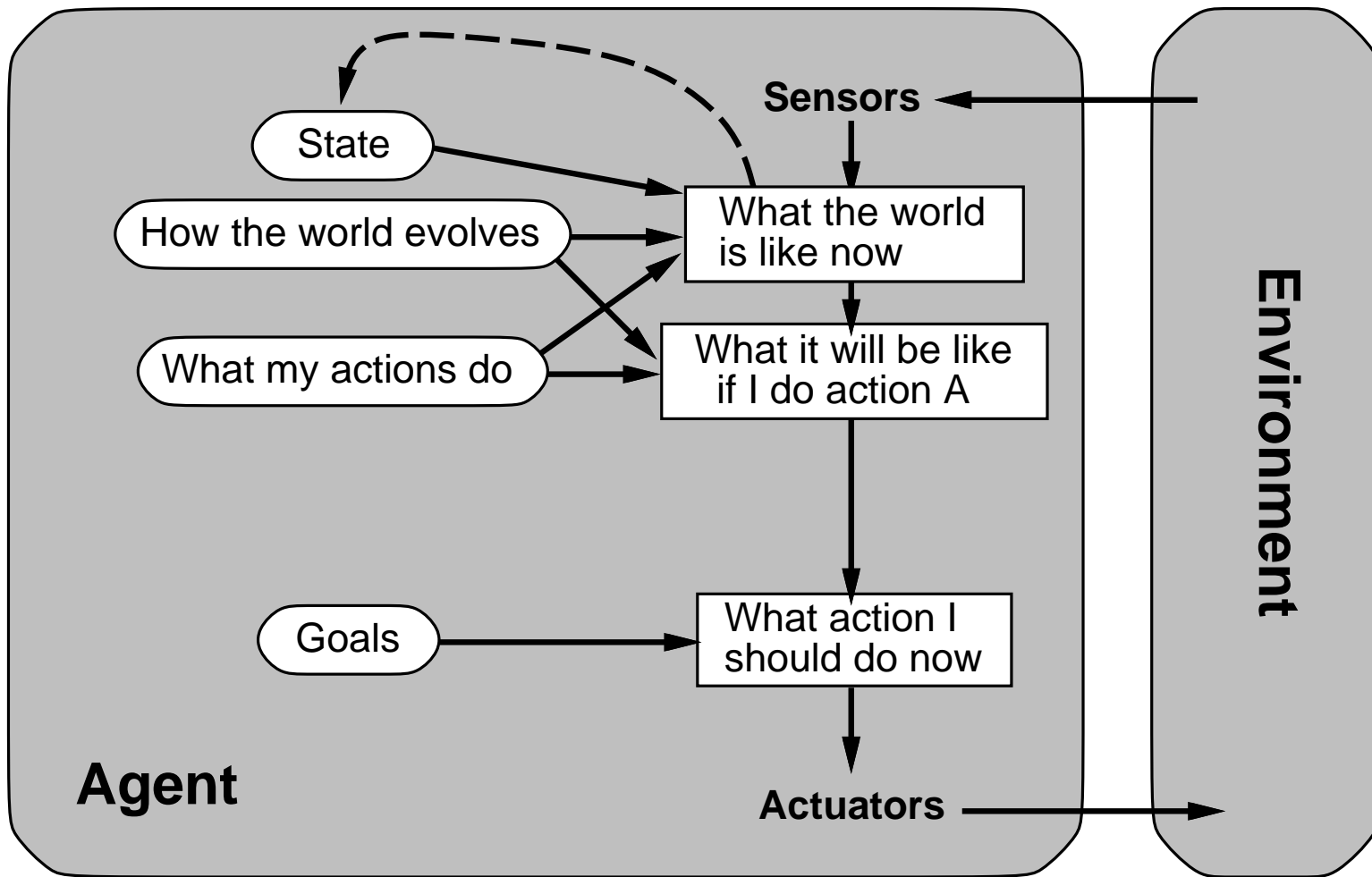
## Example

Suppose the percept didn't tell the agent what room it's in. Then the agent could remember its location:

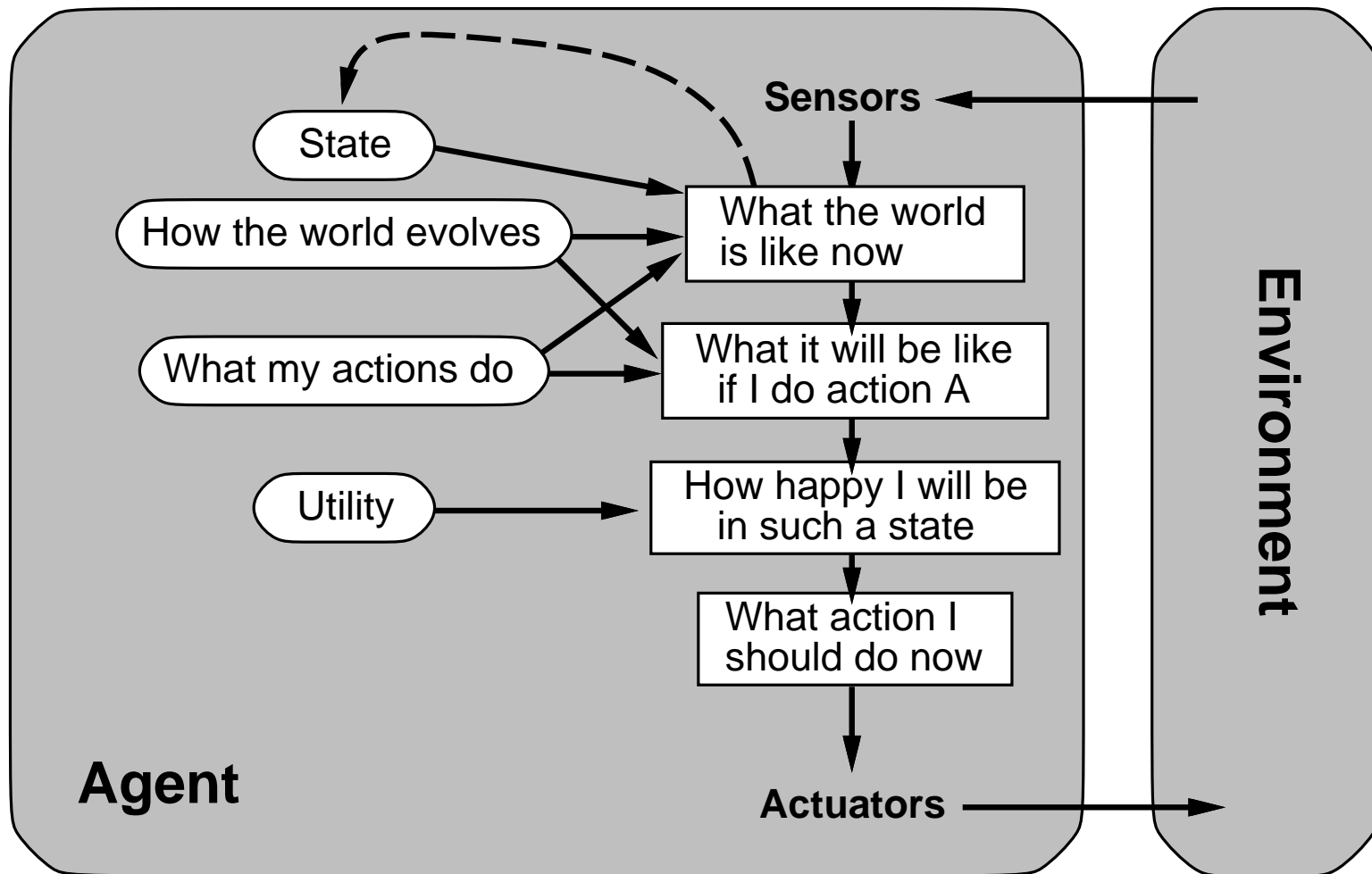
```
function VACUUM-AGENT-WITH-STATE([location]) returns an action
static: location, initially A
  if status = Dirty then return Suck
  else if location = A then
    location ← B
    return Right
  else if location = B then
    location ← A
    return Left
```

Above, I've assumed we know the agent always starts out in room *A*. What if we didn't know this?

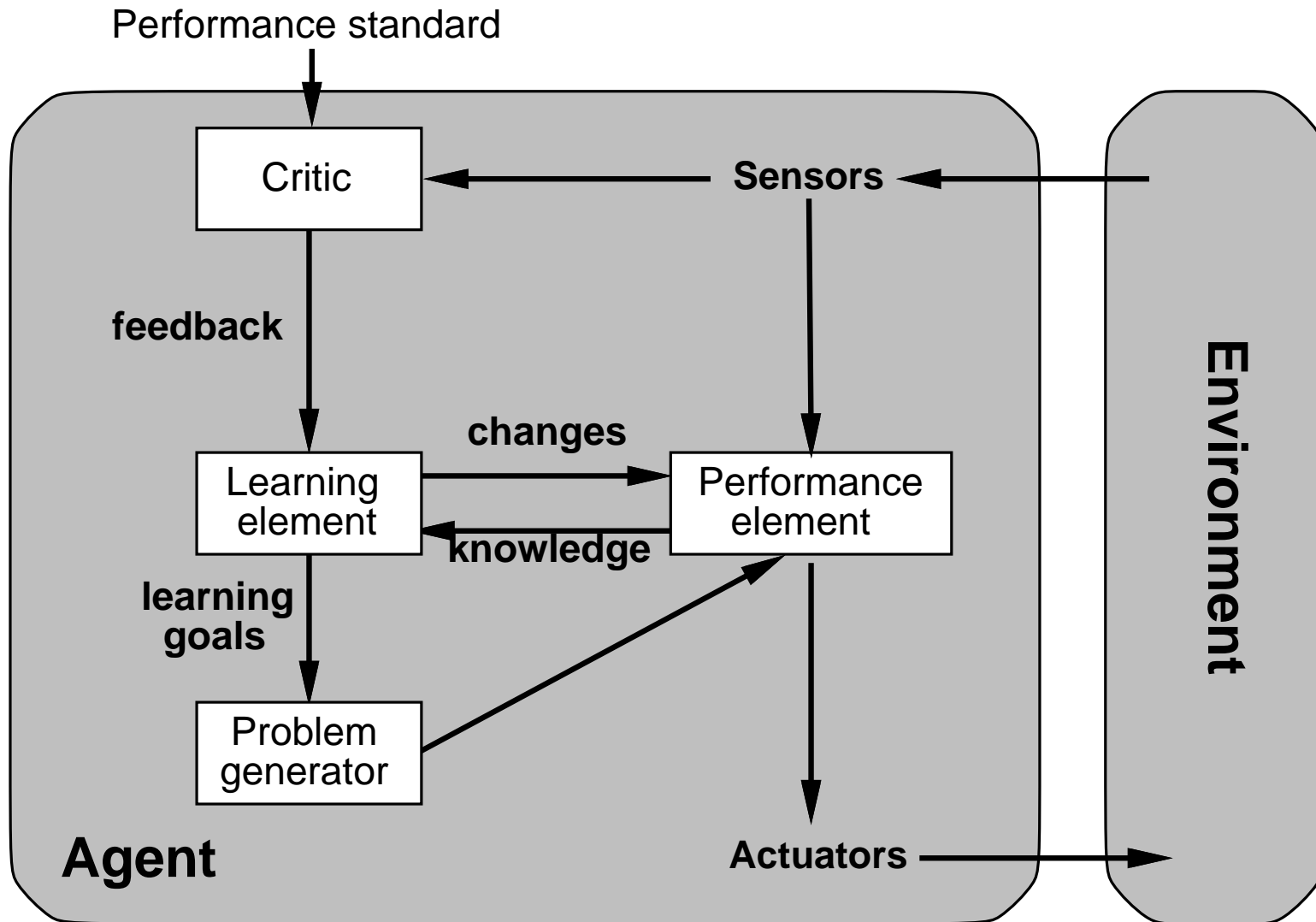
# Goal-based agents



# Utility-based agents



# 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?

Some basic agent architectures:

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