Intelligent Agents

Chapter 2

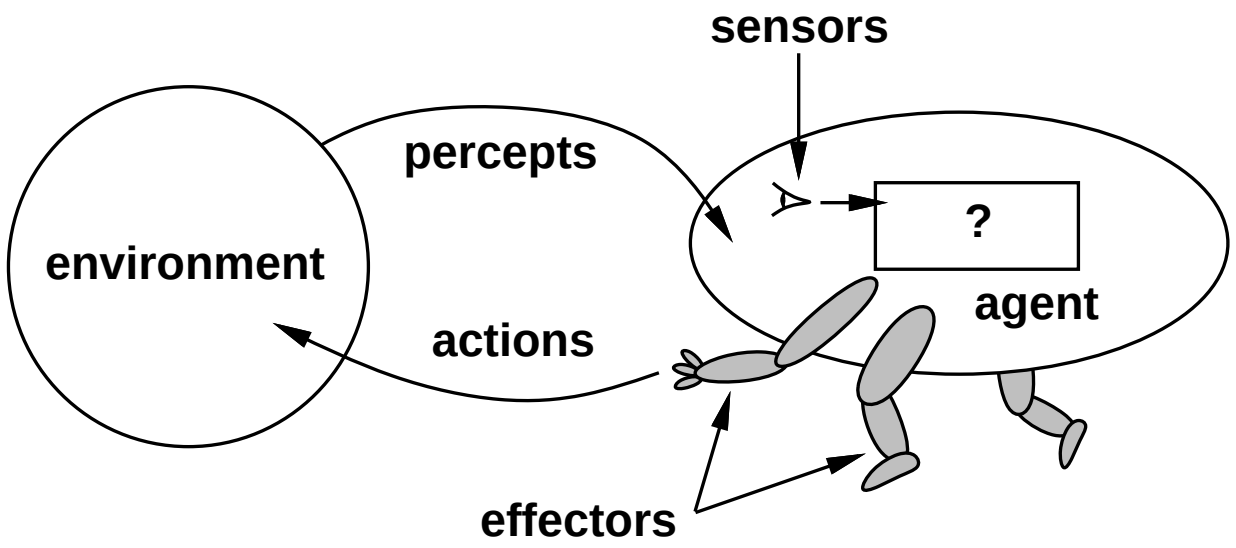
Overview

- ♢ Agent, percept, percept sequence, agent function, agent program
- ♢ Toy vacuum world
- ♢ PEAS (Performance, Environment, Actuators, Sensors) specifications
- ♢ Rationality

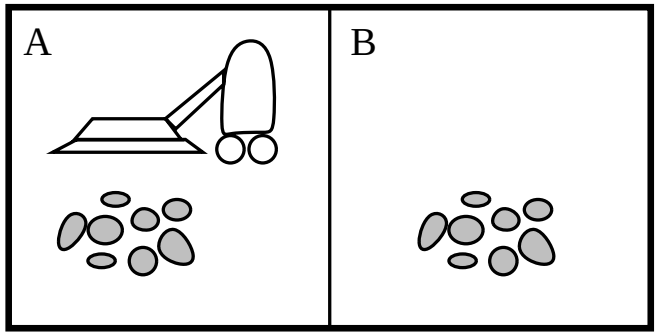
Agents

Anything that can be viewed as

- ♢ perceiving its environment through sensors and
- ♢ acting upon it through effectors/actuators



Vacuum-cleaner world



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

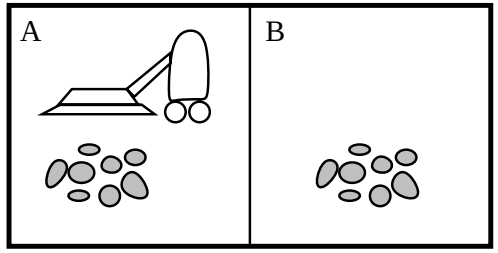
Actions: $Left$, $Right$, $Suck$, $NoOp$

Agents

♢ **Percept**: current perception of the agent

For the robot vacuum, what it sees is the location it is in and whether it's dirty.

For an autonomous car, this is what it "sees" from its sensors at the current time.



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

Agents

♢ **Percept**: current perception of the agent

♢ **Percept sequence**: complete history of everything the agent has perceived

An agent's choice of action at any moment can depend on the entire percept sequence observed to date, but not on anything it hasn't perceived.

Think about an autonomous car. The current percept may not tell you that there is a car on your blind spot, but the percept sequence would *(e.g. the fact that there was a car behind you on the back just a moment ago)*

Agents

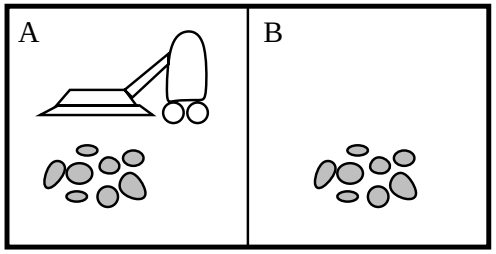
♢ **Percept**: current perception of the agent

♢ **Percept sequence**: complete history of everything the agent has perceived

♢ **Agent function**: mapping of percept sequence to actions.

We can tabulate the agent function as a percept sequence and action pairs, but that would be a very big (infinite) table unless the number of percepts and the percept length are limited.

A vacuum-cleaner agent



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

When we are designing an agent, we talk about the desired agent function as a summary ("it should stop once the rooms are cleaned etc"), but it is often not feasible to give this table.

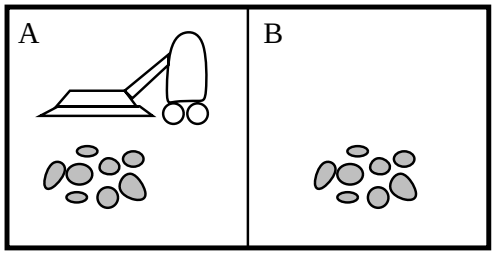
Can we write a concise **agent program**?

A vacuum-cleaner agent

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

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

One that does the *right* thing

♢ Need to define right

♢ Lets say the right thing should be roughly measured by what makes the agent most successful. Then the questions are how and when to evaluate..

Rational Agents

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

♢ **Performance measure** that defines the criterion of success

Performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.

♢ **Environment:**The agent's prior knowledge of the environment

♢ **Actions**The sensors and actions that the agent has available

♢ **Sensors:**The agent's percept sequence to date

PEAS: Perform., Envrnmnt., Actuators, Sensors

To design a rational agent, we must specify the task environment in terms of its **PEAS** description:

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

Performance measure??

Environment??

Actuators??

Sensors??

Autonomous Car

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

Consider, e.g., the task of designing an automated taxi (or autonomous car):

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?? 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)

peAS: Actuators and Sensors

Let's consider the A and S in PEAS; that is **Actuators and Sensors**

♢ The designers of the agent will decide on what hardware to put in the agent considering costs and desired functionality tasks, also depending on the environment

♢ Vacuum robot: Dirt Sensor? GPS sensor? It may be possible to do without both of these...

Peas: Performance Measure

♢ Let's consider the P in PEAS next; that is **Performance**

Performance Measure

- ♢ Amount of dirt picked up?
- ♢ Having a clean floor?
- ♢ Number of clean squares at each time step?

Performance Measure

- ♢ Amount of dirt picked up?
- ♢ Having a clean floor?
- ♢ Number of clean squares at each time step?

In general, **the performance measure should be defined as to how we want the environment when the task is completed, rather than what the agent should do.**

Rational Agents

Rational Agent Definition: 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 (environment, consequences of actions...) the agent has.

Rational Agents

Examples **explaining rationality**:

♢ Think of a poker playing agent. Let's assume its performance measure is the money it leaves the table with. The agent would be irrational if it plays not by the probabilities and game rules, but more on a whim, desires, ...

♢ Think of an autonomous car. If the agent does not look while crossing the street, can we claim that its percept sequence did not include the approaching truck, hence the agent was still rational??

No, if it has the capability, the agent is responsible in collecting the information (such as the road condition here).

In general, if you can say that with the same given (percepts and performance measure), the agent could have achieved a higher performance measure if it acted differently, the agent is not being rational/smart.

Rational Agents

Rationality is concerned with the *expected*, not necessarily the *actual* outcome

Rational $\neq$ omniscient (i.e., knows all with infinite knowledge)
percepts may not supply all relevant information.
Rational $\neq$ clairvoyant (may see future)
action outcomes may not be as expected.
Rational $\neq$ successful

Autonomy

If the actions are completely based on built-in knowledge, the agent is said to lack autonomy.

♢ A system is autonomous to the extent that its behavior is determined by its own experience

♢ Autonomy is not only it is intuitive, but it is also a sound engineering principle (what if not everything goes as expected?)

♢ Too much of it?

Just relying on own experience would be too $stupid$. Nature provides animals with enough built-in reflexes so that they survive long enough to learn for themselves

Rational+Intelligent $\Rightarrow$ exploration, learning, autonomy

Rational Agents: Summary

♢ You should understand rationality in reference to the given performance measure.

♢ You should be able to analyze the given agent function to see if it implements rational behavior for the given performance measure.

♢ You should be able to choose a performance measure that result in the desired behavior.

Properties of Environments

♢ Let's consider the E in PEAS next; that is **Environment**.

Properties of Environments

♢ **Fully vs Partially Observable**: if the sensors can detect all aspects of the environment that are relevant to the choice of action

Properties of Environments

♢ **Deterministic vs Non-deterministic**: the next state of the environment is completely determined by the current state and the agent's action.

To understand whether an environment is non-deterministic, see if there is any chance involved? I.e. the agent may make the best moves but chance may not be on its side (a good backgammon player may do all the right moves, but since backgammon is non-deterministic, his/her moves are NOT the only thing affecting the end result, or more strictly the next state.)

If an environment is not fully observable, it may $appear$ as non-deterministic. Often true for complex environments.

Note: You may generally use the terms non-deterministic and stochastic interchangeably. The book makes a small distinction (no probabilities defined in nondeterministic case) which won't be used in our discussions.

Properties of Environments

♢ **Episodic vs. Sequential**: In episodic environments, the agent receives a single percept and performs a single action, in a given episode. The next episode does not depend on the actions taken in previous episodes (e.g. defective part spotting robot decides on the quality of each part, independently of others.)

♢ **Static vs. Dynamic**: Does the environment change while an agent is deliberating

♢ **Discrete vs. Continuous**: in terms of percepts and actions

Properties of Environments

	Solitaire	Chess	Backgammon	Factory Robot	Taxi
Observable??	Partially	Fully	Yes	No	No
Deterministic??	Yes	Yes	No	No	No
Episodic??	No	No	No	Yes	No
Static??	Yes	Yes	Yes	No	No
Discrete??	Yes	Yes	Yes	No	No
Single-agent??	Yes	No	No	Maybe	Multi

Note that Solitaire is considered deterministic, because after the original ordering of the cards, there is nothing left to chance that can change/affect the success of the agent.

The environment type largely determines the agent design.

The real world is partially observable, stochastic, sequential, dynamic, continuous.

PEAS

The other dimensions of the PEAS description of the task what sensors and actuators to use) are mostly dictated by the desired agent function and costs...

Now let's look into different types of **agent programs**

Agent Programs

♢ Agent = Architecture + Program

♢ Aim: find a way to implement the rational agent function concisely

♢ An **agent program** takes a single percept as input, and returns an output action, keeping internal state.

♢ If needed for its performance measure, the agent program is responsible to keep an internal state to remember past percepts.

Table-Driven Agents

A trivial agent program: keeps track of the percept sequence and then uses it to index into a table of actions to decide what to do

```
function TABLE-DRIVEN-AGENT( percept) returns action
  static: percepts, a sequence, initially empty
          table, a table, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP( percepts, table)
  return action
```

Table-Driven Agents

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

Complexity:

Let P be the number of possible percepts and T be the total number of percepts in the sequence. The table size will be:

$$\sum_{t=1..T} P^t$$

(= the total number of possible percepts of different lengths).

Table-Driven vs. Intelligence

♢ Table-driven approach is prohibitively large even in small domains (problems in storage and creation, or even learning)!

♢ The key challenge for AI is to find how to write programs that produce rational behavior from a small code, rather than a huge table.

Ex: Square root tables being replaced with the 5-line program based on Newton's method.

Agent Types

Four basic types in order of increasing generality:

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

Simple Reflex Agents

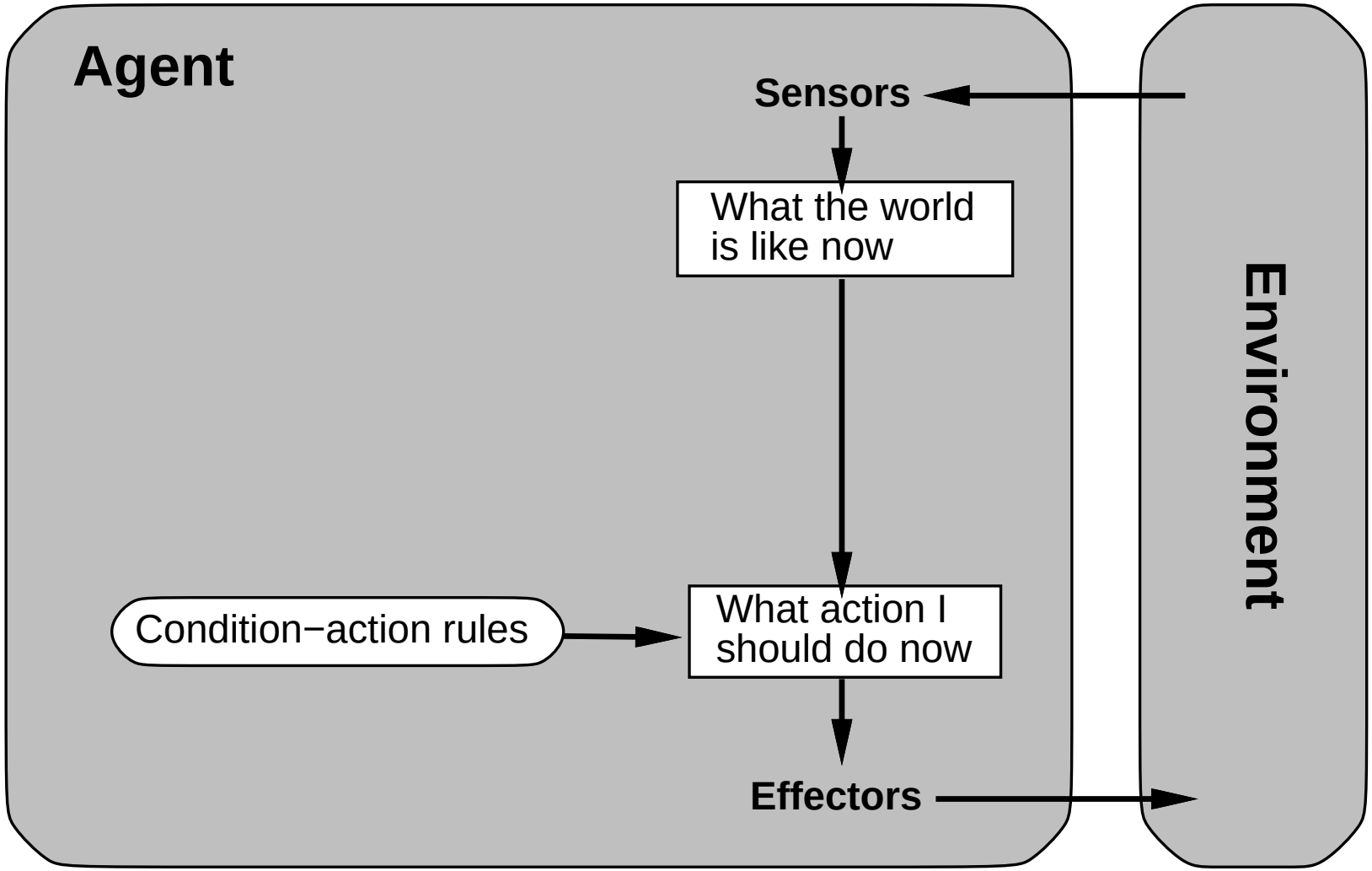
A reflex agent is like a table-driven agent that uses only the current percept.

```
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
```

ex. car driving: percept may be that the car in front is breaking; state may be that the road ahead of me is not open/free; rule may be that if road-not-free, then initiate-braking

Simple Reflex Agents



Simple Reflex Agents

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*

Advantage: very simple, short code
Disadvantage: incorrect action when when the env. is not fully observable (action depends only on the current percept!)

Reflex Agents with State

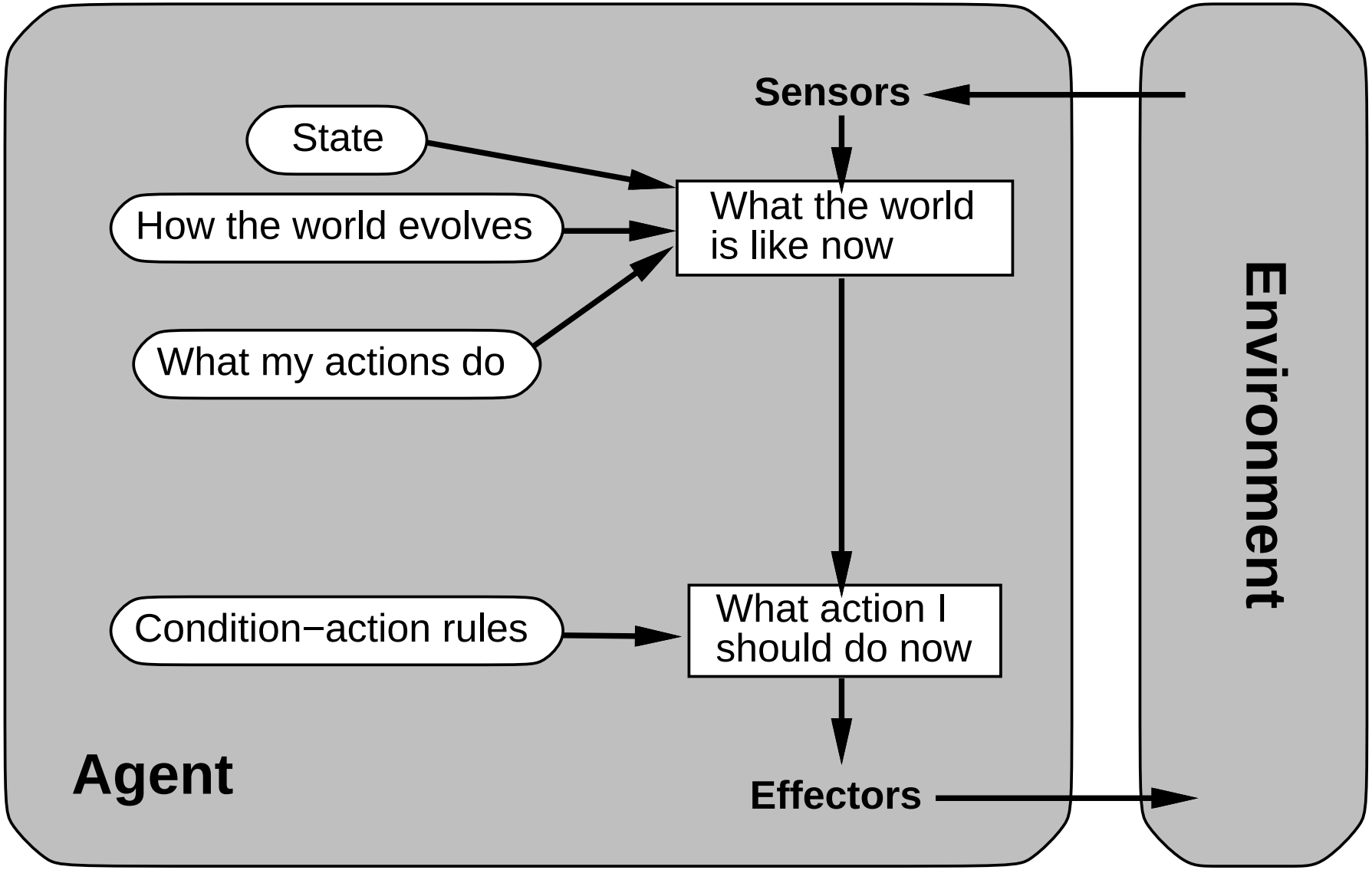
Also called **model-based agent**:

♢ Sensors do not provide access to the complete state of the world

♢ Solution: keep an internal state of the world (there was a car in the back 1 sec. ago; it was blinking towards left; ...)

We want to keep track of only the part of the world which we can't see and which is relevant.

Reflex Agents with State



Reflex Agents with State

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*

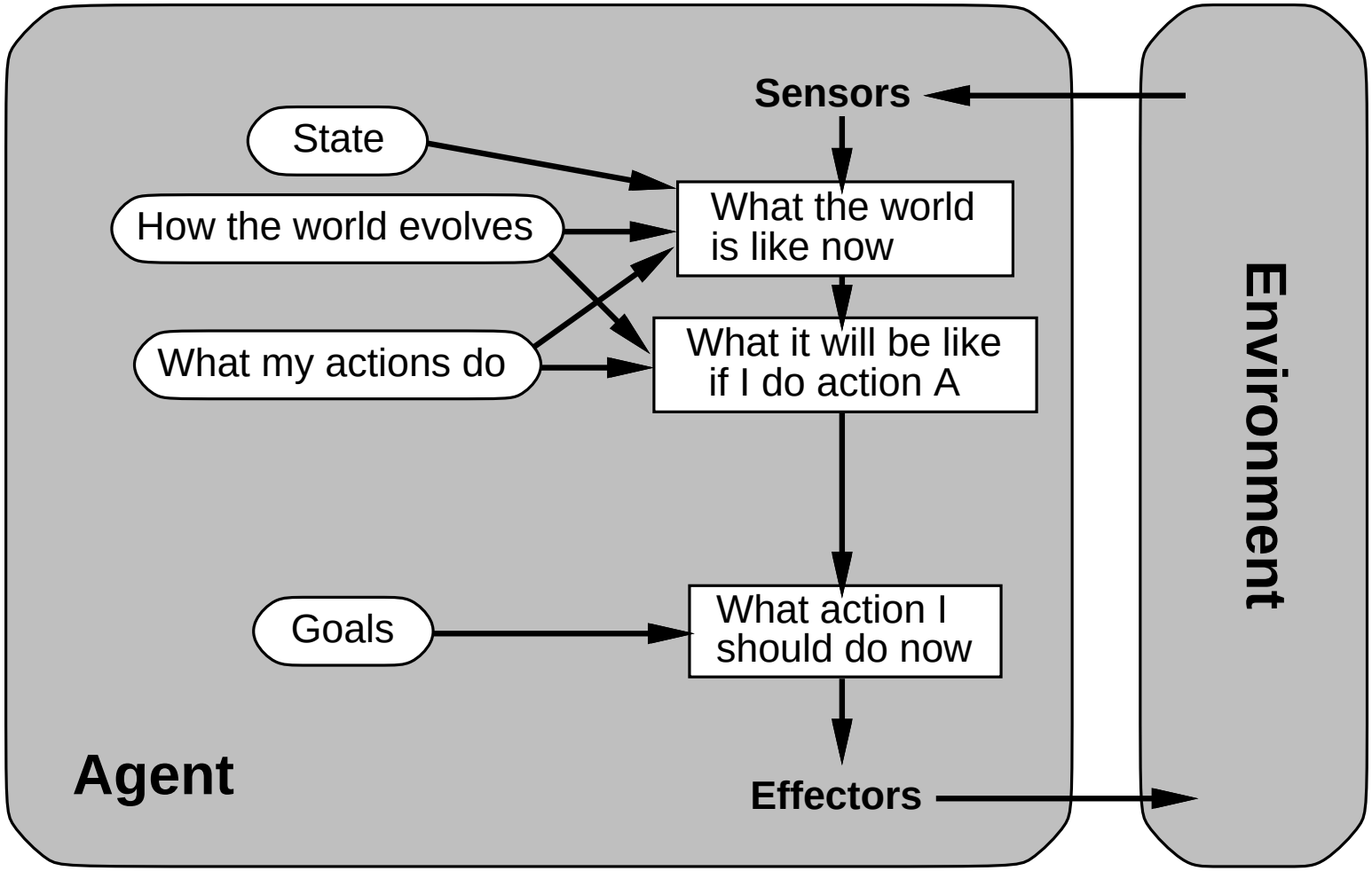
Goal-based Agents

Knowing about the current state does not always tell what to do (at a junction you can go left or right). The right decision depends on the goal.

♢ Goal-based action selection is simple when goal satisfaction results from a single action.

♢ Otherwise, selection will involve **search** and **planning**.

Goal-based Agents



Utility-based Agents

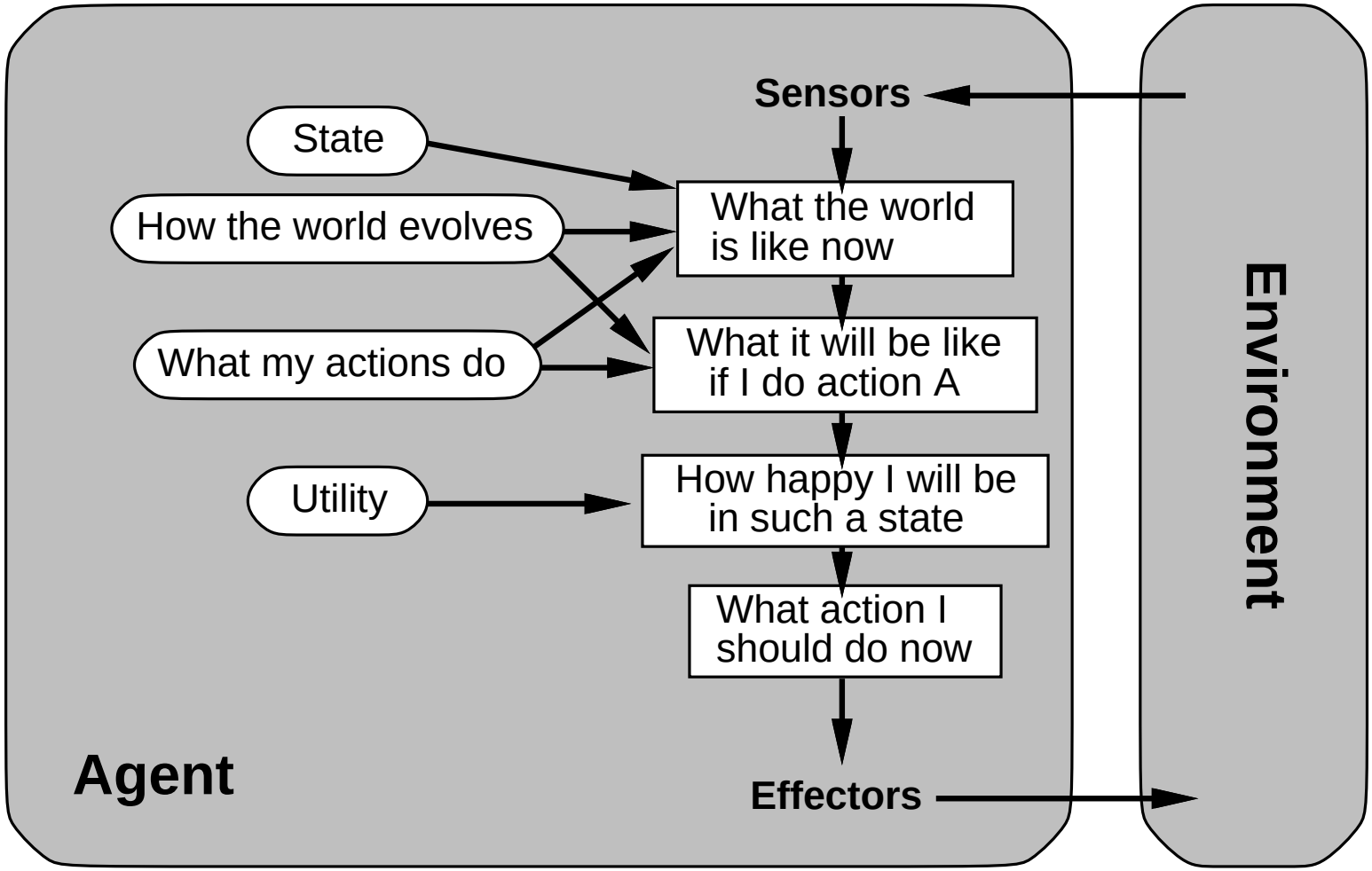
Goals alone are not enough to generate high-quality behavior (taxi getting to the target in many ways)

Utility: Maps the state to a real number

Utility functions allow rational decisions when:

♢ there are conflicting goals
♢ there is uncertainty (several goals, none of which can be achieved with certainty: must weight uncertainty with importance)

Utility-based Agents



Learning agents

So far we have described how agents select actions, but not have they *come into being.*

A **learning agent** will not need hand-programming and it will be adaptable.

What you should know

♢ Sensors, actions, environment types, percept vs percept sequence, internal state to model the world, ...

♢ Performance measure, rationality, and deciding whether an agent is rational for a given perf. measure, designing perf. measures that implement the desired behavior

♢ Different agent types (reflex agent, model-based agent, ...)