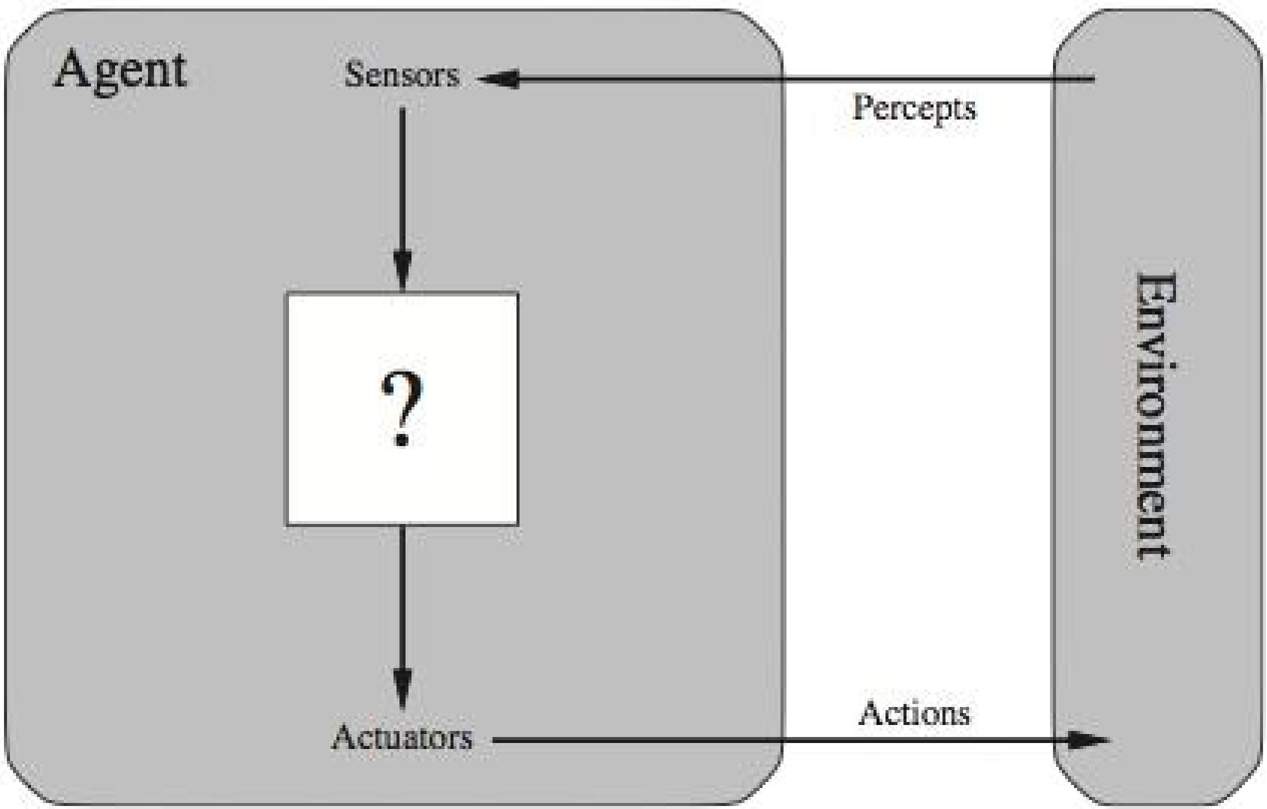
AI 2002
Artificial Intelligence

Dr. Hashim Yasin

Agent with an Environment



Task Environments

- Environment are essentially the **"problems"** to which rational agents are the **"solutions"**.

Rationality

What is rational at any given time?

It depends on four things:

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

Task Environments

- In ***designing an agent***, the first step must always be to specify the task environment (PEAS) as fully as possible.

PEAS:

  - Performance measure
  - Environment
  - Actuators
  - Sensors

Task Environments … Examples

PEAS for an automated taxi-driving

- **Performance measure:** Safe, fast, legal, comfortable trip, maximize profits, etc.
- **Environment:** Roads, other traffic, pedestrians, customers, etc.
- **Actuators:** Steering wheel, accelerator, brakes, signal, horn, etc.
- **Sensors:** Cameras, speedometer, GPS, odometer, engine sensors, keyboard, etc.

Task Environments ... Examples

PEAS for medical diagnosis system

- **Performance measure:** Healthy patient, minimize costs etc.
- **Environment:** Patient, hospital, staff etc.
- **Actuators:** Screen display (questions, tests, diagnoses, treatments, referrals)
- **Sensors:** Keyboard (entry of symptoms, findings, patient's answers)

Task Environments ... Examples

PEAS for satellite image analysis system

- **Performance measure:** correct image categorization
- **Environment:** downlink from the orbiting satellite
- **Actuators:** display categorization of scene
- **Sensors:** colour pixel arrays

Environment Types

Environment Types

- **Fully observable** vs. **partially observable**
- **Deterministic** vs. **stochastic**
- **Episodic** vs. **sequential**
- **Static** vs. **dynamic**
- **Discrete** vs. **continuous**
- **Single agent** vs. **multi-agent**

Environment Types

Fully observable vs. partially observable

Fully observable:

- If an agent's sensors give it access to the ***complete*** state of the environment at each point in time.
- A task environment is effectively fully observable if *the sensors detect all aspects that are relevant to the choice of action*.
- Convenient, because the agent need not to maintain any internal state to keep track of the world

Partially observable:

- ***Parts*** *of the state are simply missing* from sensor data
- Noisy and inaccurate sensors
  - A vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares
  - An automated taxi cannot see what other drivers are thinking.

Environment Types

Deterministic vs. stochastic

Deterministic:

- If the next state of the environment is completely determined by the current state and the action executed by the agent.
- Vacuum-cleaner world is deterministic.

Stochastic:

- If the environment is partially observable then it could be stochastic.
- Taxi driving is clearly stochastic in this sense, because one can never predict the behaviour of the traffic exactly.

Environment Types

Episodic vs. **Sequential**

Episodic:

- In episodic environments, the agent's experience is divided into atomic episodes.
  - Each episode consists of the agent perceiving and then performing a single action.
  - The next episode does not depend on the actions taken in previous episodes.
- Example is the classification tasks

Sequential:

- In sequential environments, the current decision could affect all future decisions.
- Examples are Chess and taxi driving

Environment Types

Static vs. **Dynamic**

Static:

- Static environments are unchanged and easy to deal with because the agent need not keep looking at the world while it is deciding on an action.
- Crossword puzzles are static.

Dynamic:

- If the environment can be changed while an agent is deliberating, then we say the environment is dynamic for that agent --- taxi driving

- *If the environment itself does not change with the passage of time but the agent's performance score changes*, then we say the environment is **semi-dynamic** --- Chess when played with a clock, is semi-dynamic

Environment Types

Discrete vs. Continuous

- The discrete/continuous distinction can be applied to the **state of the environment**, to the way **time** is handled, and to the **percept** and **actions**.
  - **Chess** has a discrete set of percept and actions.
  - **Taxi driving** contains a continuous state and continuous-time problem,
  - **Taxi-driving** actions are also continuous.

Environment Types

Single agent vs. **multi-agent**

- An agent operating by itself in an environment is a single agent.
- Examples:
  - Crossword puzzle -> a single agent
  - chess -> two-agents.

Does an agent "A" have to treat an object "B" as an agent, or can it be treated merely as a stochastically behaving object

- The key distinction is whether B's behaviour is best described as maximizing a performance measure whose value depends on agent A's behaviour.

Agent functions and Programs

- **Agent program**

  takes the current percept as an input from the sensors and returns an action to the actuators.
- **Agent function**

  takes the whole percept history and maps onto actions.
- Notice the difference between the agent program, which takes the current percept as input, and the agent function, which takes the entire percept history.
- The agent needs to remember the whole percept sequence, if it needs it.

Table-driven Agent

Table-driven Agent Program

- 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** 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 ← LOOKUP(*percepts*, *table*)
  return *action*

Percept sequence	Actions
[A,Clean]	Right
[A, Dirty]	Suck
[B,Clean]	Left
[B,Dirty]	Suck
[A,Clean],[A,Clean]	Right
[A,Clean],[A,Dirty]	Suck
…	…
[A,Clean],[A,Clean],[A,Clean]	Right
[A,Clean],[A,Clean],[A, Dirty]	Suck

Table-driven Agent

Why the table-driven approach for agent construction is considered as failure.

- The lookup table will contain the **number of entries**

$$\sum_{t=1}^{\mathrm{T}} |\mathcal{P}|^t$$

Where,

- $\mathcal{P}$ is the set of percept
- T is the lifetime

Table-driven Agent

Example 1: Automated taxi:

- The visual input from a single camera comes in at the rate of roughly **27 megabytes per second** with info
  - 30 frames per second,
  - 640 x 480 pixels with 24 bits of colour information
- This gives a lookup table with over $10^{250,000,000,000}$ entries for an hour's driving.

Example 2: Chess:

- Even the lookup table for chess—a tiny, well-behaved real world—would have at least 10^{150} entries.

Agent Types

Agent Types

- There are following four kinds of agent
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

Simple Reflex Agents

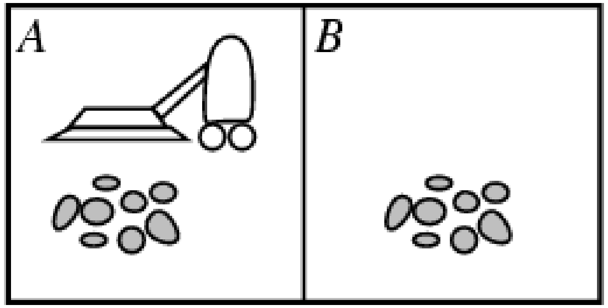
- Select actions on the basis of the **current percept** and ignoring the rest of the percept history
- **Condition-action rule**

  if *car-in-front-is-braking* **then** *initiate-braking*.

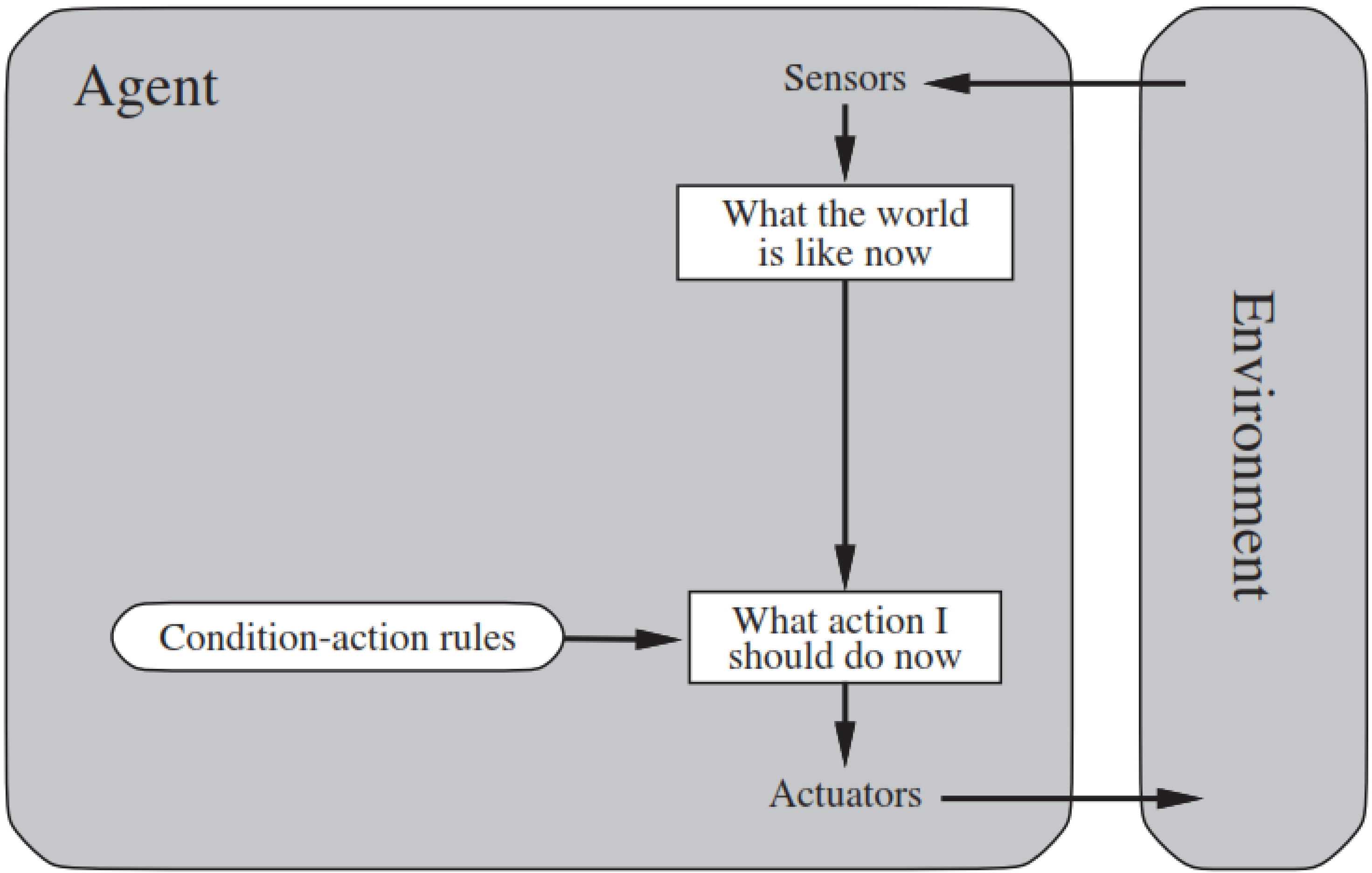
- **Vaccum Cleaner World:**

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** $Left$



Simple Reflex Agents



Simple Reflex Agents

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

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

- The agent will work only if
  - the **correct** decision can be made on the basis of the current percept –that is, only if the environment is **fully observable.**

Model-based Reflex Agents

Model:

- *A description that how the next state depends on the current state & action.*

- It handles partial observability in a more effective way.

- It maintains some sort of **internal state** that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.

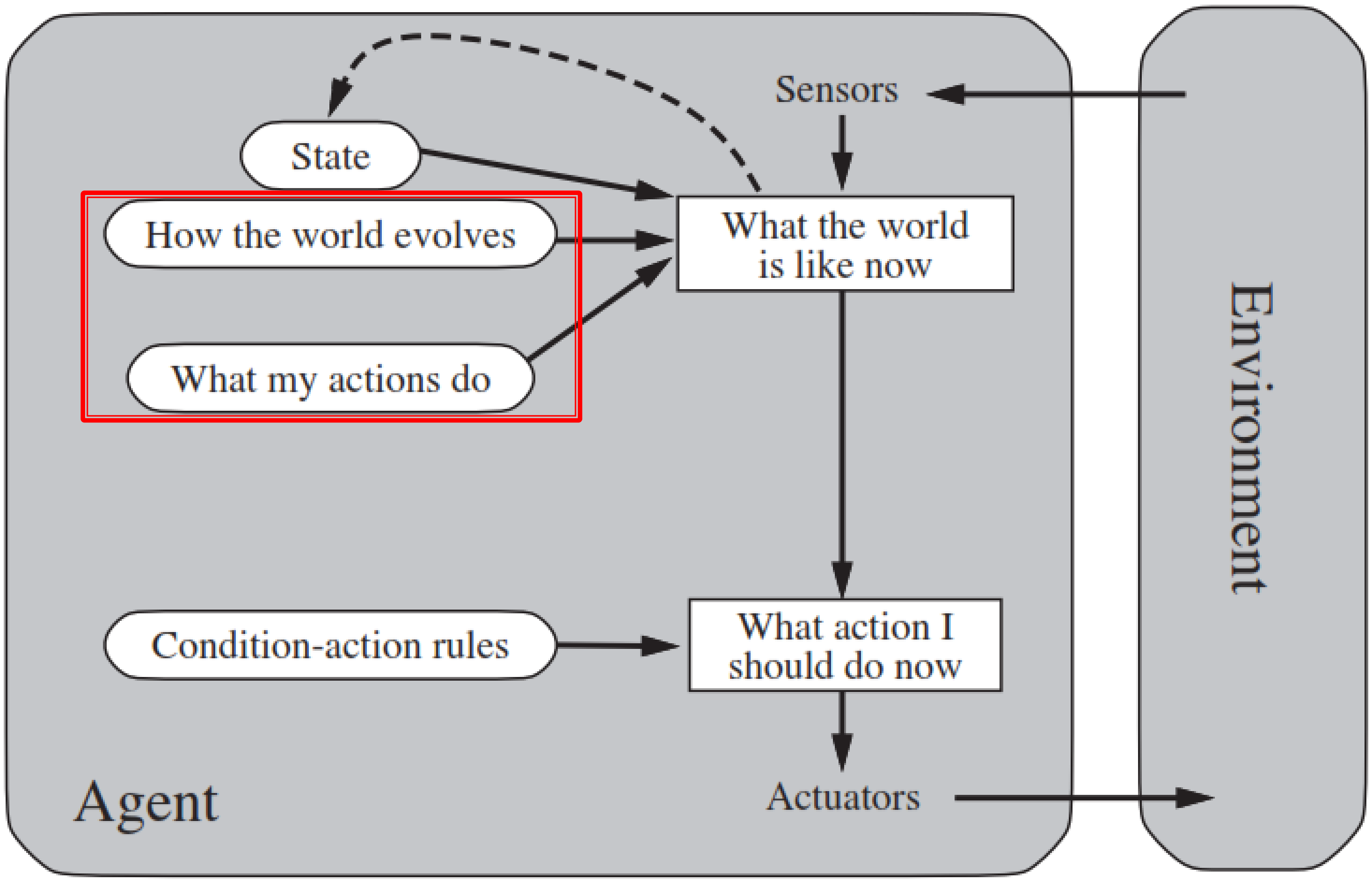
Model-based Reflex Agents

- Updating this internal state information requires **two kinds of knowledge**:

  - **First,** how the world evolves independently of the agent.

  - **Second,** how the agent's own actions affect the world.

Model-based Reflex Agents



Model-based Reflex Agents

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    model, a description of how the next state depends on current state and action
    rules, a set of condition–action rules
    action, the most recent action, initially none

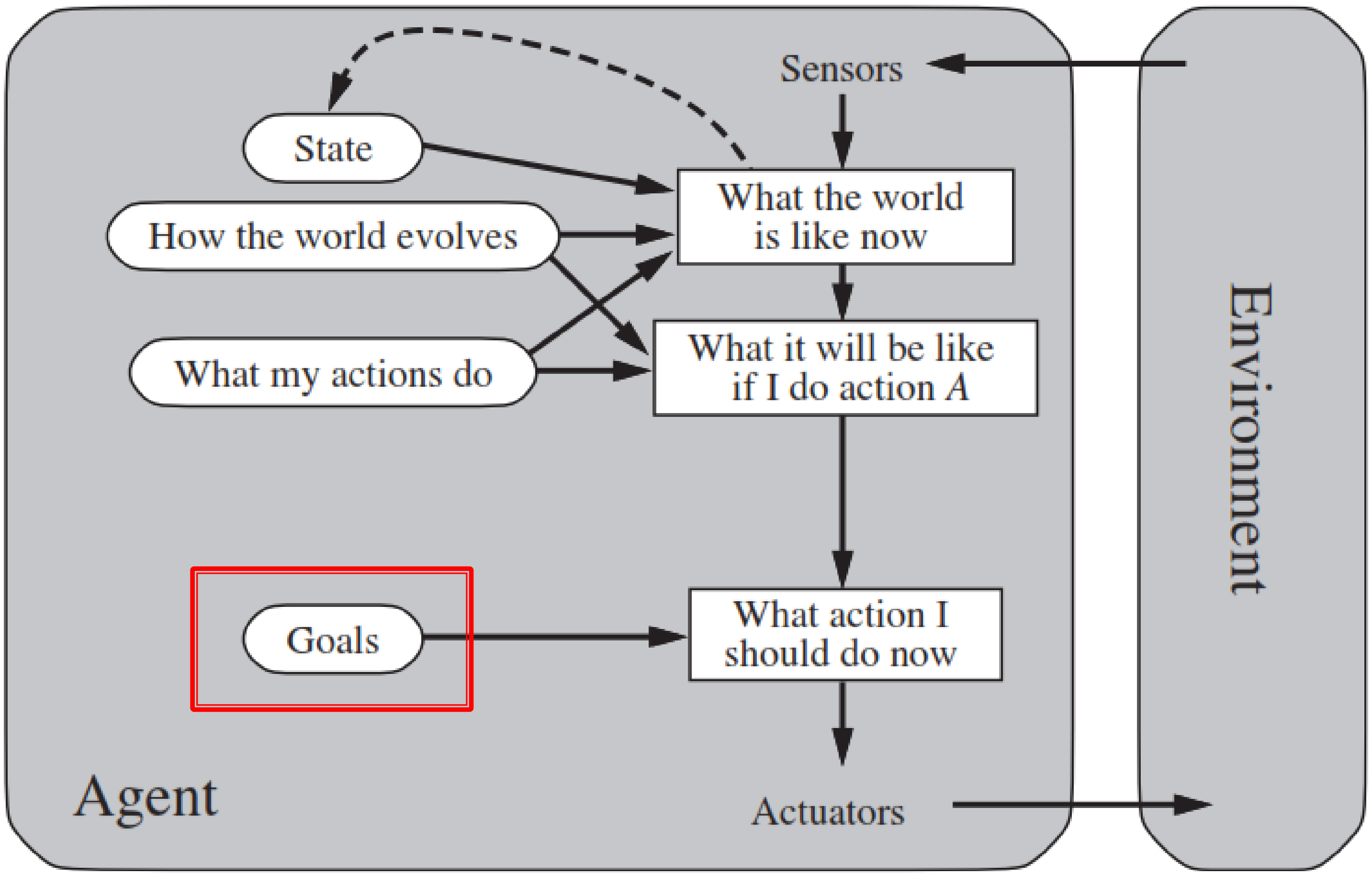
  state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

The **Model** is the knowledge about "how the world works".

Goal-based Agents

- Information about the current state of the environment is not always enough to decide what to do (e.g. decision at a road junction).
- The agent needs some sort of **goal information** that describes situations that are desirable.
- The agent program can combine this with information about the results of possible actions in order to choose actions that achieve the goal.
- Usually requires some **search and planning**.

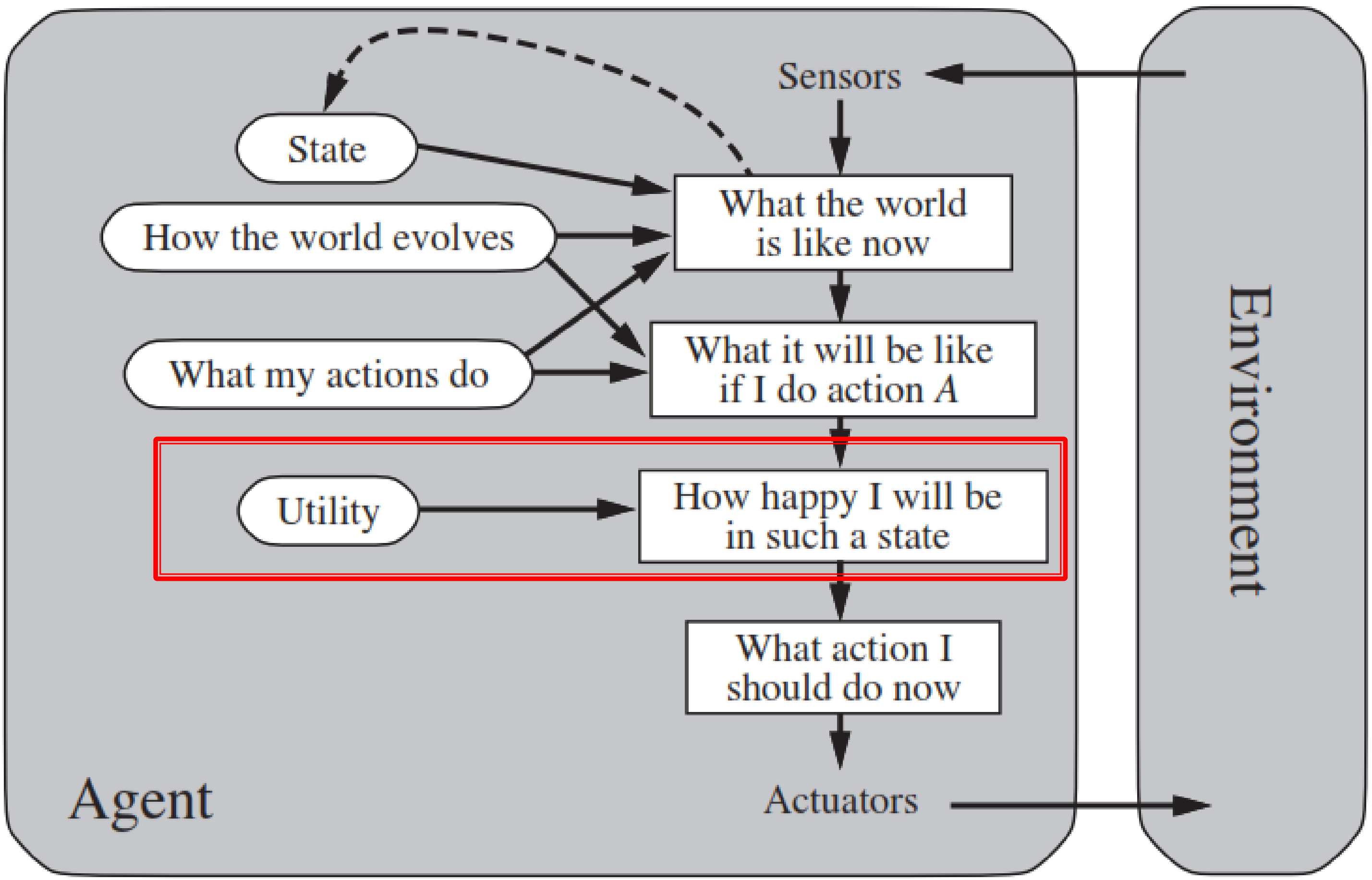
Goal-based Agents



Utility-based Agents

- Goals provide a **binary distinction between happy and unhappy states.**
- Agents so far we have discussed had a single goal. Agents may have to *juggle conflicting goals*.
- Need to optimise utility over a range of goals.
- **Utility**: measure of *happiness (a real number), --- the quality of being useful .*
- A **utility function** *maps a state onto a real number which describes the associated degree of happiness.*

Utility-based Agents



Reading Material

- **Artificial Intelligence,** A Modern Approach

  Stuart J. Russell and Peter Norvig

  - **Chapter 2.**