Intelligent Agents

Chapter 2

Outline

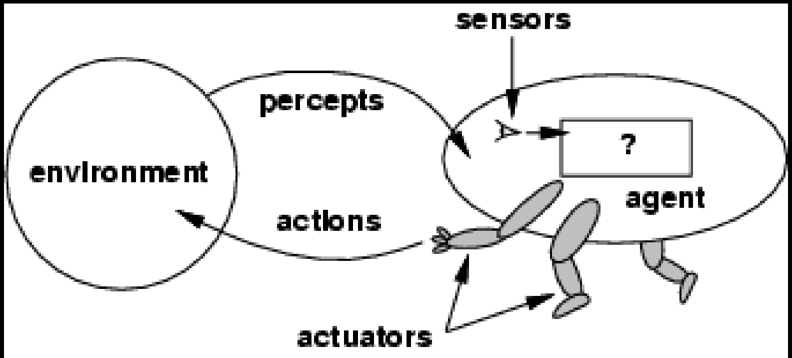
- Agents and environments
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators. "Amazon Go"
- 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

percepts and **actuators** that convert commands into

Agents and environments



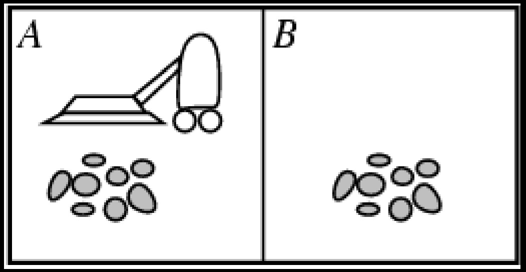
The agent function maps from percept histories to actions:

$$[f: \mathcal{P}^{\star} \rightarrow \mathcal{A}]$$

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

- \input{tables/vacuum-agent-function-table}
-

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 is the one that will cause the agent to be most successful
- Performance measure: An objective criterion for success of an agent's behavior
- E.g., 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.
-

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.
-

An agent acts **intelligently** when

- what it does is appropriate for its circumstances and its goals,
- it is flexible to changing environments and changing goals,
- it learns from experience, and
- it makes appropriate choices given its perceptual and computational limitations. An agent typically cannot observe the state of the world directly; it has only a finite memory and it does not have unlimited time to act.

A **computational** agent is an agent whose decisions about its actions can be explained in terms of computation. That is, the decision can be broken down into primitive operation that can be implemented in a physical device.

Knowledge is the information about a domain that can be used to solve problems in that domain.

Rational agents

- 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

- PEAS: Performance measure, Environment, Actuators, Sensors
- Must first specify the setting for intelligent agent design
-
- Consider, e.g., the task of designing an automated taxi driver:
-
  - Performance measure
  - Environment
  - Actuators
  - Sensors

PEAS

- Must first specify the setting for intelligent agent design
-
- Consider, e.g., the task of designing an automated 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)

PEAS

- 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

PEAS

- Agent: Interactive English tutor
- Performance measure: Maximize student's score on test
- Environment: Set of students
- Actuators: Screen display (exercises, suggestions, corrections)
- Sensors: Keyboard

The four application domains are as follows:

An **autonomous delivery robot** roams around a building delivering pack-ages and coffee to people in the building. This delivery agent should be able to find paths, allocate resources, receive requests from people, make decisions about priorities, and deliver packages without injuring people or itself.

A **diagnostic assistant** helps a human troubleshoot problems and suggests repairs or treatments to rectify the problems.

A **tutoring system** interacts with a student, presenting information about some domain and giving tests of the student’s knowledge or performance.

A **trading agent** knows what a person wants and can buy goods and services on her behalf.

An Autonomous Delivery Robot

Imagine a robot that has wheels and can pick up objects and put them down. It has sensing capabilities so that it can recognize the objects that it must manipulate and can avoid obstacles.

autonomous delivery robot has the following as inputs:

Previous knowledge, provided by the agent designer, about its own capabilities, what objects it may encounter and have to differentiate, what requests mean, and perhaps about its **environment**, such as a **map**;

past experience obtained while acting, for instance, about the effect of its actions, what objects are common in the world, and what requests to expect at different times of the day;

goals in terms of what it should deliver and when, as well as preferences that specify trade-offs, such as when it must forgo one goal to pursue another, or the trade-off between acting quickly and acting safely; and

observations about its environment from such input devices as cameras, sonar, touch, sound, laser range finders, or keyboards.

Agent functions and programs

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

Table-lookup agent

- \input{algorithms/table-agent-algorithm}
-
- Drawbacks:
  - Huge table
  - Take a long time to build the table
  - No autonomy
  - Even with learning, need a long time to learn the table entries

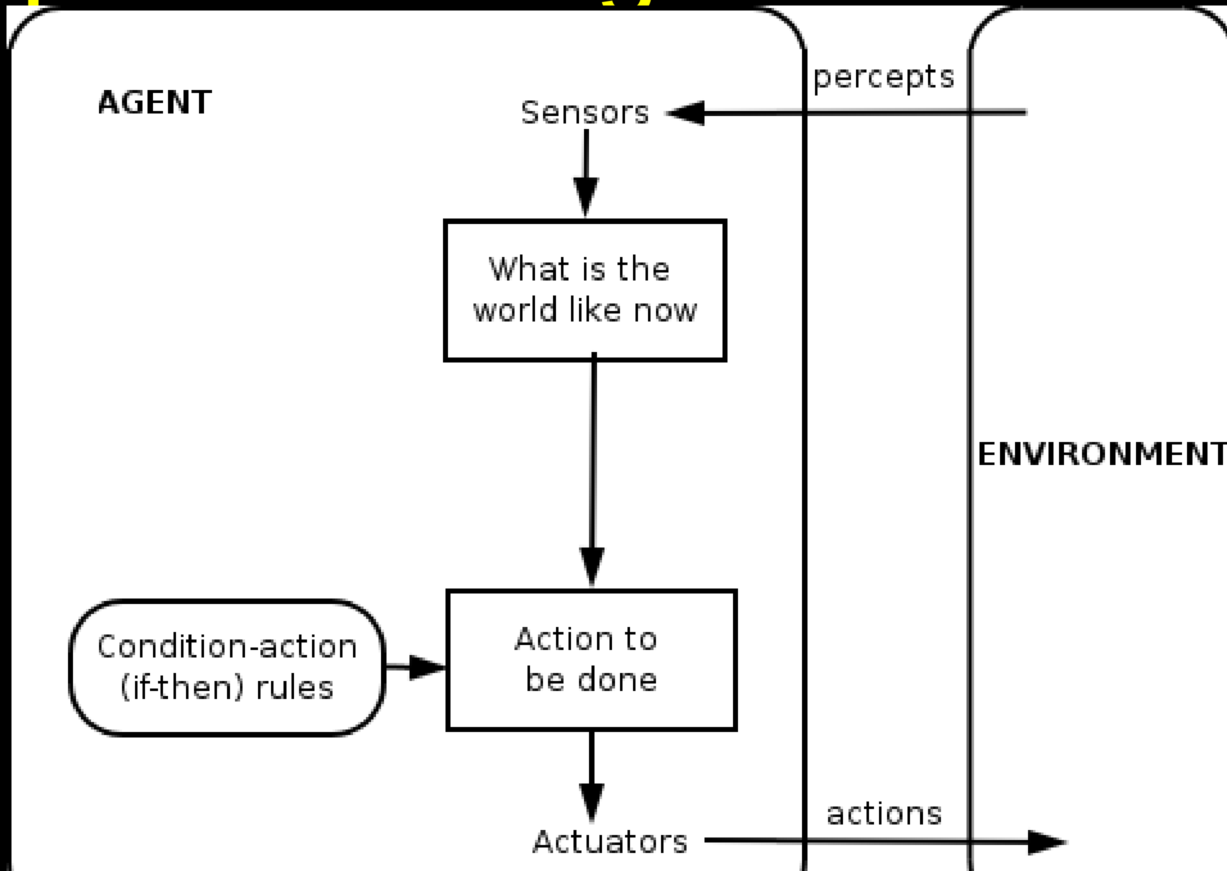
Agent program for a vacuum-cleaner agent

- \input{algorithms/reflex-vacuum-agent-algorithm}
-

Agent types

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

Simple reflex agents



A simp… hes the

curren…

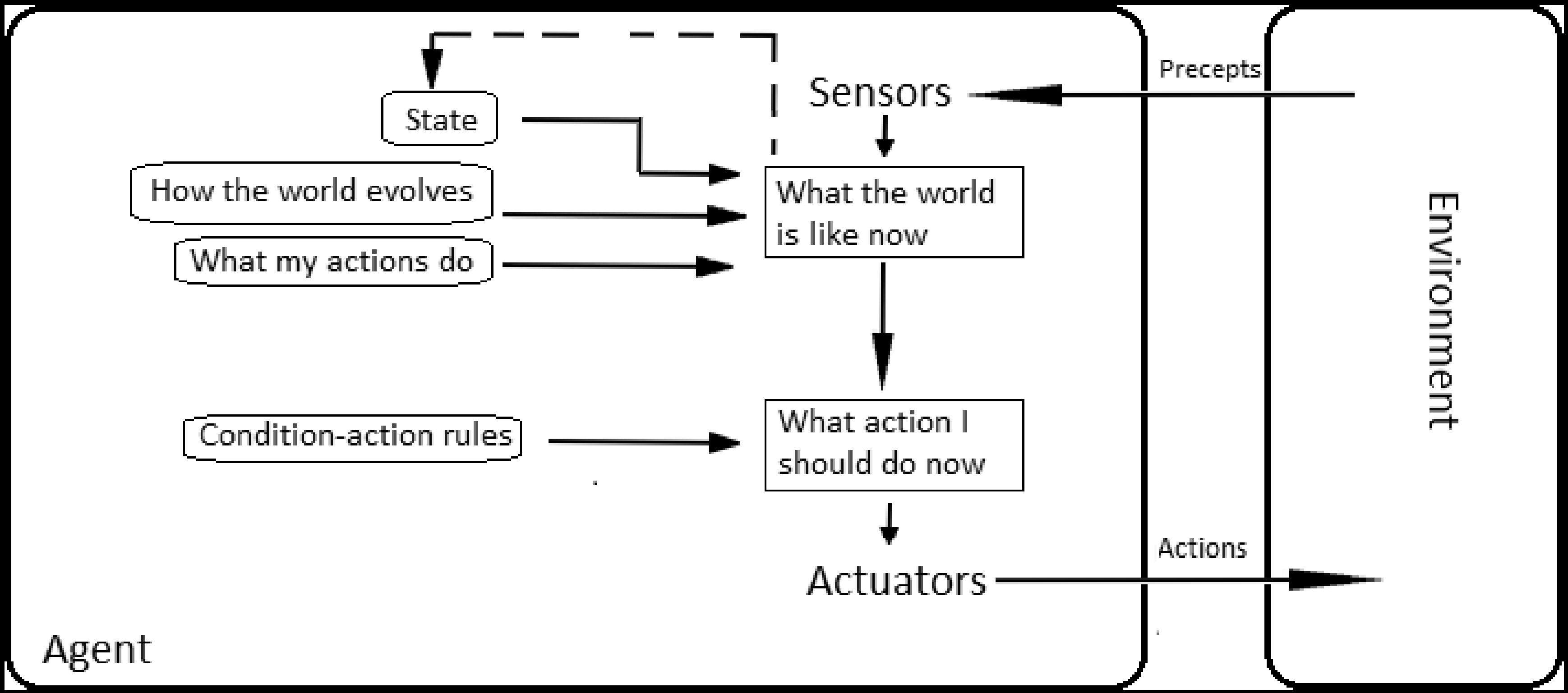
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action

persistent: *rules*, a set of condition–action rules

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

return *action*

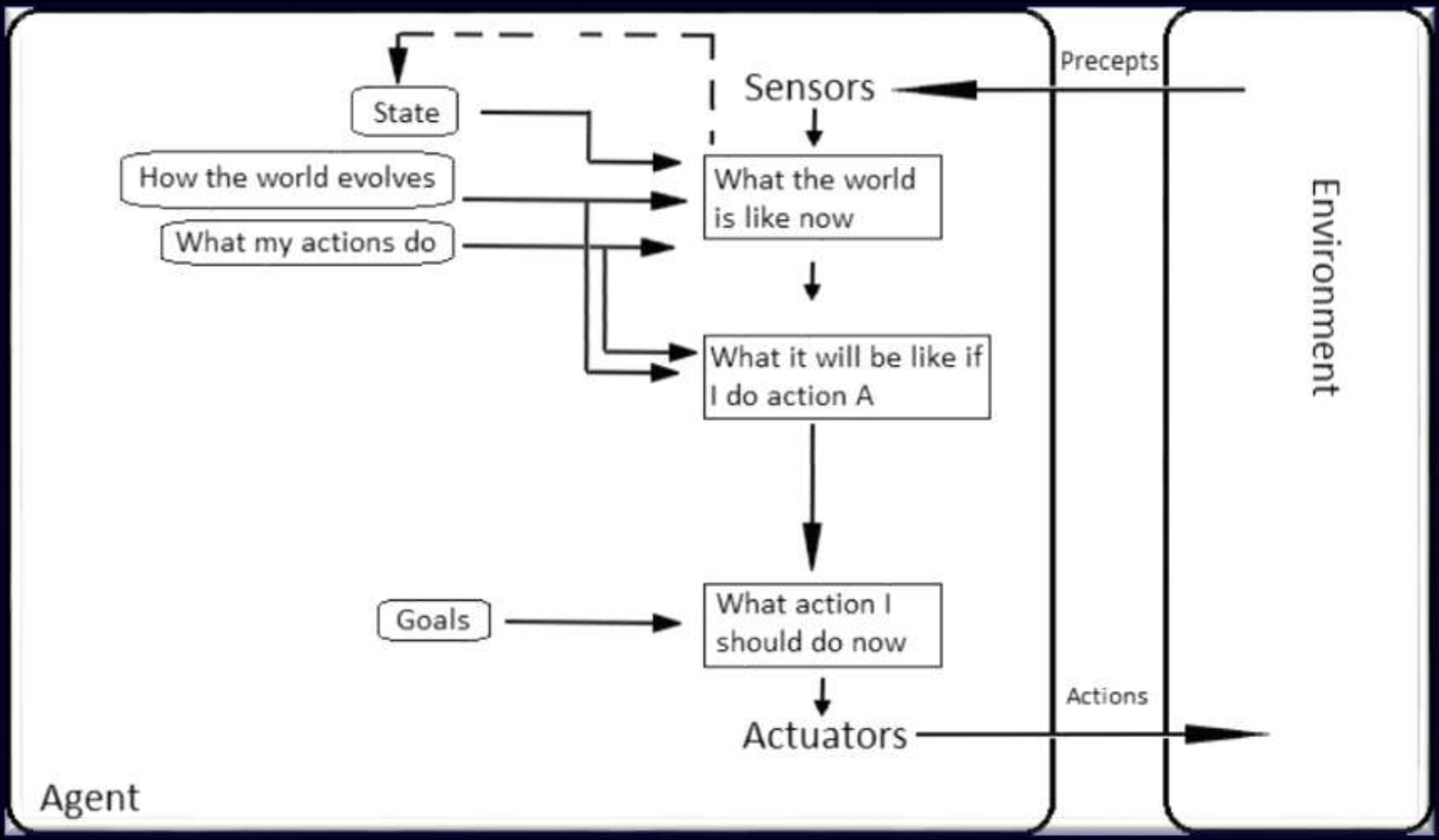
Model-based reflex agents



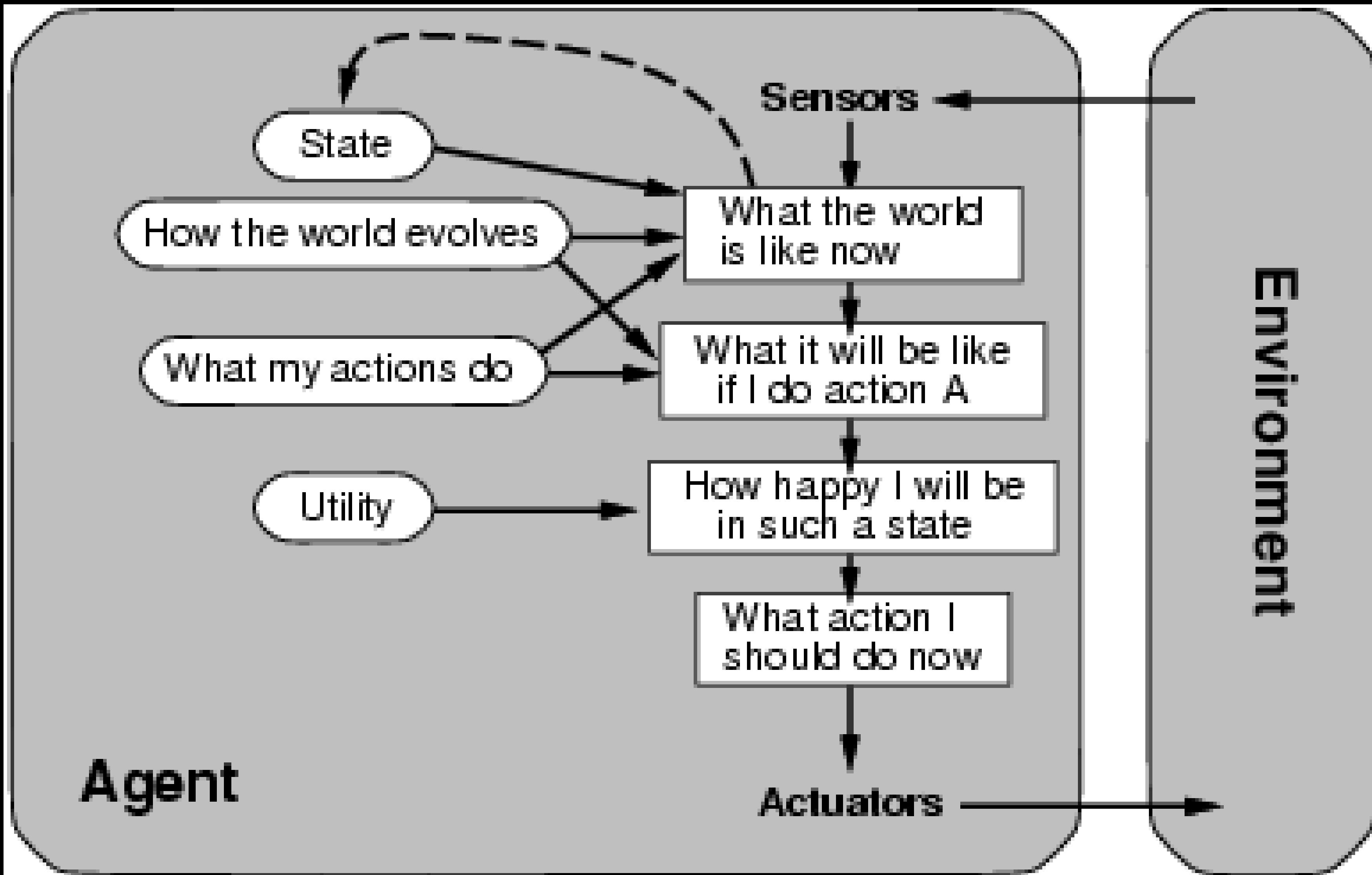
the agent should maintain some sort of **internal state** that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.

For the **braking** problem, the internal state is not too extensive— just the previous frame from the camera, allowing the agent to detect when two red lights at the edge of the vehicle go on or off simultaneously. For other driving tasks such as changing lanes, the agent needs to keep track of where the other cars are if it can't see them all at once. And for any driving to be possible at all, the agent needs to keep track of where its keys are.

Goal-based agents



Utility-based agents



Learning agents