Intelligent Agents

Bùi Tiến Lên

01/09/2019

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Contents

Agents and Environments

What is Agent

Concept 1

An **agent** is anything that can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators**

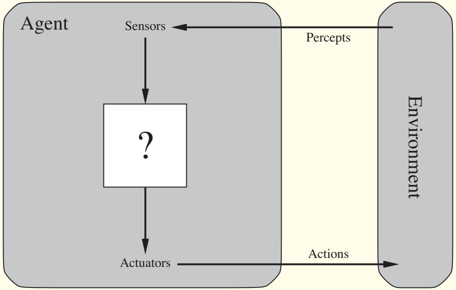
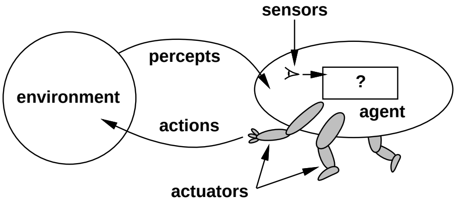


Figure 1: Agents interact with environments through sensors and actuators.

What is Agent (cont.)

- **Agents** can be humans, robots, softbots, thermostats, etc.



What is Agent (cont.)

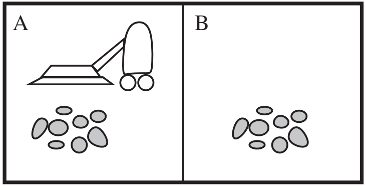
- **Percept** to refer to the agent's perceptual inputs at any given instant.
- An agent's **percept sequence** is the complete history of everything the agent has ever perceived.
- An agent's **behavior** is described by the **agent function** that maps any given percept sequence to an action

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

- **Agent program**: the implementation of the agent function

$$agent = architecture + program$$

Vacuum-cleaner



- Percepts: location and contents, e.g., $[A, \mathit{Dirty}]$
- Actions: *Left*, *Right*, *Suck*, *NoOp*
- Function:

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

Vacuum-cleaner (cont.)

```
function REFLEX-VACCUM-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?

Good Behavior: The Concept of Rationality

Rationality

- A **rational agent** is one that does the **right thing**
- What is **right thing**?
  - When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives. This sequence of actions causes the environment to go through a sequence of states. If the sequence is desirable, then the agent has performed well
  - **Performance measure** evaluates any given sequence of **environment states** (not **agent states**)

Rationality (cont.)

A rule of thumb

It is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave

Rationality (cont.)

Rational 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.

Rationality (cont.)

Concept 2 (Definition of a 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.

Omniscience, learning, and autonomy

- 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 $\implies$ exploration, learning, autonomy

Omniscience vs. Rationality

Omniscience	**Rationality**
Knows the actual outcome of its actions in advance	Rationality maximizes *expected* performance, while perfection maximizes *actual* performance
Perfection but not practical	

Omniscience, learning, and autonomy (cont.)

Information gathering

- **Information gathering** by doing actions in order to modify future percepts or exploration
- This is an important part of rationality

Omniscience, learning, and autonomy (cont.)

Learning

- A rational agent also has to **learn** as much as possible from what it perceives.
  - The agent's initial configuration may be modified and augmented as it gains experience.
- There are extreme cases in which the environment is completely known **a priori**.

Omniscience, learning, and autonomy (cont.)

Autonomy

- A rational agent should be **autonomous** – Learn what it can to compensate for partial or incorrect prior knowledge.
  - If an agent just relies on the prior knowledge of its designer rather than its own percepts then the agent lacks autonomy

The Nature of Environments

The task environment

- **Task environments** are essentially the “problems” to which rational agents are the “solutions”
  - The flavor of the task environment directly affects the appropriate design for the agent program
- **Task environment** includes the **PEAS** (**P**erformance, **E**nvironment, **A**ctuators, **S**ensors) description
- In designing an agent, the first step must always be to specify the task environment as fully as possible.

An example: Automated taxi driver

- **Performance measure**
  - safety, destination, profits, legality, comfort, ...
- **Environment**
  - streets/freeways, traffic, pedestrians, weather, ...
- **Actuators**
  - steering, accelerator, brake, horn, speaker/display, ...
- **Sensors**
  - video, accelerometers, gauges, engine sensors, keyboard, GPS, ...

Software agents

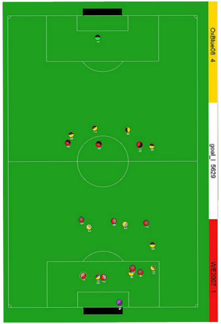
- Sometimes, the environment may not be the real world.
  - Flight simulator, video games, Internet
  - They are all artificial but very complex environments
- Those agents working in these environments are called **software agents** (**softbots**).
  - All parts of the agent are software.

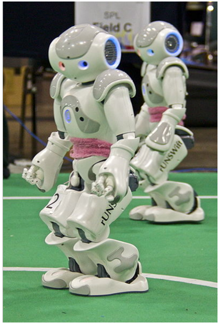
Properties of task environments

- Fully observable vs. partially observable
- Single agent vs. multiagent
- Deterministic vs. stochastic
- Episodic vs. sequential
- Discrete vs. continuous
- Static vs. dynamic
- Known vs. unknown

Fully observable vs. partially observable

- Fully observable: The agent's sensory gives it access to the complete state of the environment.
  - The agent need not maintain internal state to keep track of the world.
- Partially observable
  - Noisy and inaccurate sensors
  - Parts of the state are simply missing from the sensor data
- Unobservable: The agent has no sensors at all



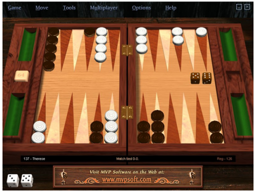


Single agent vs. multiagent

- Single agent: An agent operates by itself in an environment.
  - Solving crossword → single agent, playing chess → two agents
- Which entities must be viewed as agents?
- Competitive vs. Cooperative multiagent environment
  - Playing chess → competitive, driving on road → cooperative

Deterministic vs. stochastic

- Deterministic: The next state of the environment is completely determined by the current state and the action executed by the agent.
  - The vacuum world → deterministic, driving on road → stochastic
- Most real situations are so complex that they must be treated as stochastic.



Episodic vs. sequential

- Episodic: The agent's experience is divided into atomic episodes, in each of which the agent receives a percept and then performs a single action
- Sequential: A current decision could affect future decisions

Discrete vs. continuous

- The discrete/continuous distinction applies to the *state* of the environment, to the way *time* is handled, and to the *percepts* and *actions* of the agent

The other properities

Static vs. dynamic

- **Static**: The environment is unchanged while an agent is deliberating.
  - Crossword puzzles $\rightarrow$ static, taxi driving $\rightarrow$ dynamic
- **Semidynamic**: The environment itself does not change with the passage of time but the agent's performance score does
  - Chess playing with a clock

Known vs. unknown

- **Known environment**: the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given.
- **Unknown environment**: the agent needs to learn how it works to make good decisions.

The Structure of Agents

The architecture of agents

- We have formula

$$agent = architecture + program$$

- Architecture: some sort of computing device with physical sensors and actuators that this program will run on.
  - Ordinary PC, robotic car with several onboard computers, cameras, and other sensors, etc.
- The program has to be appropriate for the architecture.
  - Program: Walk action $\rightarrow$ Architecture: legs

The agent programs

- A trivial agent program: keep track of the percept sequence and 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
```

The agent programs (cont.)

- The table-driven approach to agent construction is doomed to failure
- Let P be the set of possible percepts and
- Let T be the lifetime of the agent (the total number of percepts it will receive).
- The lookup table will contain $\sum_{t=1}^{T} |P|^t$ entries $\rightarrow$ very huge table

Agent types

Four basic types in order of increasing generality:

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

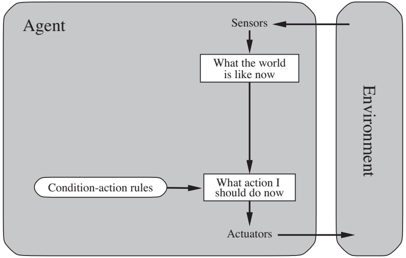
Simple reflex agents

- The simplest kind of agent, but of limited intelligence
- Select actions based on the current percept, ignoring the rest of the percept history
- The connection from percept to action is represented by condition-action rules.

IF *current percept* **THEN** *action*

- Limitations
  - Knowledge sometimes cannot be stated explicitly $\rightarrow$ low applicability
  - Work only if the environment is fully observable

Simple reflex agents (cont.)

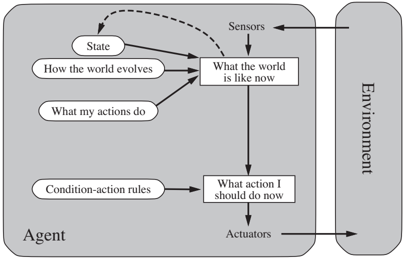


function SIMPLE-REFLEX-AGENT(*percept*)
returns an action
persistent: *rules*, a set of condition-action rules
  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

Model-based reflex agents

- Partially observability $\rightarrow$ the agent has to keep track of an internal state
  - Depend on the percept history and reflect some of the unobserved aspects
- The agent program updates the internal state information as time goes by by encoding two kinds of knowledge
  - How the world evolves independently of the agent
  - How the agent's actions affect the world

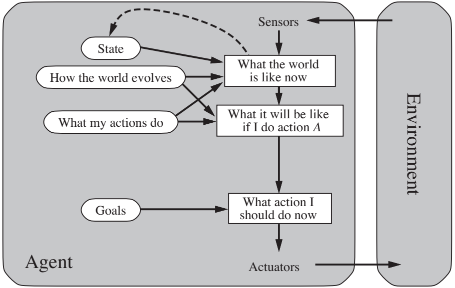
Model-based reflex agents (cont.)




```
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
```

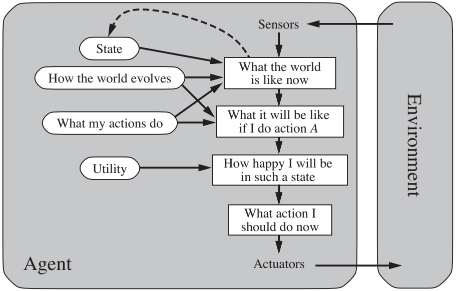
Goal-based agents

- Current state of the environment is always not enough
- The agent further needs some sort of goal information that describes situations that are desirable.
- Less efficient but more flexible



Utility-based agents

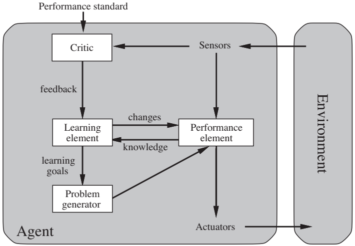
- Goals alone are not enough to generate high-quality behavior in most environments
- Many action sequences to get the goals, some are better and some worse
- An agent's **utility function** is essentially an internalization of the performance measure.
  - Goal $\rightarrow$ success, utility $\rightarrow$ degree of success (how successful it is)



Learning agents

A learning agent is divided into four conceptual components

- **Learning element** $\rightarrow$ Making improvement
- **Performance element** $\rightarrow$ Selecting external actions
- **Critic** $\rightarrow$ Tells the Learning element how well the agent is doing with respect to fixed performance standard. (Feedback from user or examples, good or not?)
- **Problem generator** $\rightarrow$ Suggest actions that will lead to new and informative experiences



Component representations

- Three basic representations: **atomic**, **factored**, and **structured**

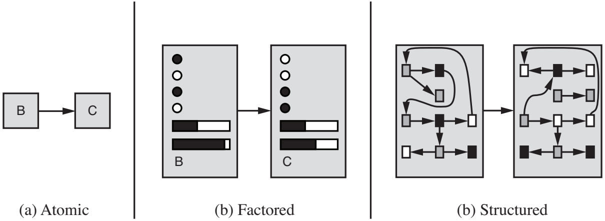


Figure 2: Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols. (c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

References

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep learning.
MIT press.

Nguyen, T. (2018).
Artificial intelligence slides.
Technical report, HCMC University of Sciences.

Russell, S. and Norvig, P. (2016).
Artificial intelligence: a modern approach.
Pearson Education Limited.