Intelligent Agents

Chapter 2
Agents

Anything that can be viewed as

◊ perceiving its environment through sensors and
◊ acting upon it through effectors/actuators
Agents

◊ Percept: current perception

◊ Percept sequence: complete history of everything the agent has perceived

◊ Agent function: mapping of percept sequence to actions

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.

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 is limited.
Percepts: location sensor and dirt sensor, e.g., \([A, Dirty]\)

Actions: \(Left, Right, Suck, NoOp\)
## A vacuum-cleaner agent

<table>
<thead>
<tr>
<th>Percept sequence</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>([A, \text{Clean}])</td>
<td>(\text{Right})</td>
</tr>
<tr>
<td>([A, \text{Dirty}])</td>
<td>(\text{Suck})</td>
</tr>
<tr>
<td>([B, \text{Clean}])</td>
<td>(\text{Left})</td>
</tr>
<tr>
<td>([B, \text{Dirty}])</td>
<td>(\text{Suck})</td>
</tr>
<tr>
<td>([A, \text{Clean}], [A, \text{Clean}])</td>
<td>(\text{Right})</td>
</tr>
<tr>
<td>([A, \text{Clean}], [A, \text{Dirty}])</td>
<td>(\text{Suck})</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
A vacuum-cleaner agent

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

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...

◊ Performance measure: criteria that determine how successful an agent is (of course this is not a fixed measure, it will change according to tasks)

◊ Vacuum cleaner robot - how? when?
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**
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??** 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)
Properties of Environments

◊ Let's consider the E in PEAS first; that is Environment.

◊ We will later look at the Performance measure
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

<table>
<thead>
<tr>
<th></th>
<th>Solitaire</th>
<th>Chess</th>
<th>Backgammon</th>
<th>Factory</th>
<th>Robot</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Observable??</strong></td>
<td>Partially</td>
<td>Fully</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Deterministic??</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Episodic??</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Static??</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Discrete??</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Single-agent??</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Maybe</td>
<td>Multi</td>
<td></td>
</tr>
</tbody>
</table>

The environment type largely determines the agent design.

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

If you can say that with the same given (percepts, knowledge and performance measure), the agent could have achieved a higher score if it acted differently, the agent is not being rational.
Rational Agents

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).

We should design our agents to be search for info in order to make good decisions:

♦ Information gathering: Collecting information in order to obtain better future percepts

♦ Learning
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 $\neq$ 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 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 ⇒ exploration, learning, autonomy
Structure of Intelligent Agents

Agent = Architecture + Program

Hardware/Software agents (what can you say about the two environments complexity?)
Agent Programs

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.

We had said before that the agent program takes the whole percept sequence to date, so the agent program is responsible to keep an internal state to remember past percepts.


**Table-Driven Agents**

```plaintext
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
```

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

=\* Prohibitively large even in small domains (problems in storage and creation, learning).
Table-Driven vs. Intelligence

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 (interpreting percepts, rules: if car-in-front-brakes then initiate-braking)
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 the env. is not fully observable
(action depends only on the current percept!)
Reflex Agents with State

Also called model-based reflex agent:

◊ Sensors do not provide access to the complete state of the world (blind spot)

◊ 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

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.
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)
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.