Foundations of Artificial Intelligence

2. Rational Agents
Nature and Structure of Rational Agents and Their Environments

Wolfram Burgard, Bernhard Nebel, and Martin Riedmiller

Albert-Ludwigs-Universität Freiburg

April 27, 2012
1. What is an agent?
2. What is a rational agent?
3. The structure of rational agents
4. Different classes of agents
5. Types of environments
Agents

- Perceive the environment through sensors (→ Percepts)
- Act upon the environment through actuators (→ Actions)

Examples: Humans and animals, robots and software agents (softbots), temperature control, ABS, ...
Rational Agents

... do the “right thing”!

In order to evaluate their performance, we have to define a performance measure.

Autonomous vacuum cleaner example:
- $m^2$ per hour
- Level of cleanliness
- Energy usage
- Noise level
- Safety (behavior towards hamsters/small children)

Optimal behavior is often unattainable
- Not all relevant information is perceivable
- Complexity of the problem is too high
Rationality vs. Omniscience

- An *omniscient agent* knows the *actual effects* of its *actions*

- In comparison, a *rational agent* behaves according to its *percepts* and *knowledge* and attempts to *maximize the expected performance*

- Example: If I look both ways before crossing the street, and then as I cross I am hit by a meteorite, I can hardly be accused of lacking rationality.
The Ideal Rational Agent

Rational behavior is dependent on

- Performance measures (goals)
- Percept sequences
- Knowledge of the environment
- Possible actions

Ideal 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.

Active perception is necessary to avoid trivialization. The ideal rational agent acts according to the function

$$\text{Percept Sequence} \times \text{World Knowledge} \rightarrow \text{Action}$$
# Examples of Rational Agents

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Performance Measure</th>
<th>Environment</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medical diagnosis system</td>
<td>healthy patient, costs, lawsuits</td>
<td>patient, hospital, stuff</td>
<td>display questions, tests, diagnoses, treatments, referrals</td>
<td>keyboard entry of symptoms, findings, patient’s answers</td>
</tr>
<tr>
<td>Satellite image analysis system</td>
<td>correct image categorization</td>
<td>downlink from orbiting satellite</td>
<td>display categorization of scene</td>
<td>color pixel arrays</td>
</tr>
<tr>
<td>Part-picking robot</td>
<td>percentage of parts in correct bins</td>
<td>conveyor belt with parts, bins</td>
<td>jointed arm and hand</td>
<td>camera, joint angle sensors</td>
</tr>
<tr>
<td>Refinery controller</td>
<td>purity, yield, safety</td>
<td>refinery, operators</td>
<td>valves pumps, pumps, heaters displays</td>
<td>temperature, pressure, chemical sensors</td>
</tr>
<tr>
<td>Interactive English tutor</td>
<td>student’s score on test</td>
<td>set of students, testing agency</td>
<td>display exercises, suggestions, corrections</td>
<td>keyboard entry</td>
</tr>
</tbody>
</table>

(University of Freiburg) Foundations of AI April 27, 2012
Structure of Rational Agents

Realization of the ideal mapping through an

- Agent program, executed on an
- Architecture which also provides an interface to the environment (percepts, actions)

→ Agent = Architecture + Program
function TABLE-DRIVEN-AGENT(\textit{percept}) \textbf{returns} an action
\begin{itemize}
\item persistent: \textit{percepts}, a sequence, initially empty
\item \textit{table}, a table of actions, indexed by percept sequences, initially fully specified
\end{itemize}
append \textit{percept} to the end of \textit{percepts}
\begin{itemize}
\item \textit{action} \leftarrow \text{LOOKUP}(\textit{percepts}, \textit{table})
\end{itemize}
\textbf{return} \textit{action}

Problems:
\begin{itemize}
\item The table can become very large
\item and it usually takes a very long time for the designer to specify it (or to learn it)
\item \ldots practically impossible
\end{itemize}
Direct use of perceptions is often not possible due to the large space required to store them (e.g., video images).

Input therefore is often interpreted before decisions are made.
Interpretative Reflex Agents

Since storage space required for perceptions is too large, direct interpretation of perceptions is not feasible. Therefore, an alternative approach is needed. In lieu of direct interpretation, an algorithmic function is employed to convert perceptions into actions. This function is named `TABLE-D-RIVEN-AGENT` and is defined as follows:

```plaintext
function TABLE-D-RIVEN-AGENT(percept) returns an action
persistent: percepts, a sequence, initially empty
action ← LOOKUP(percepts, table)
return action
```

The `TABLE-D-RIVEN-AGENT` function is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

---

**Figure 2.3**
The `TABLE-D-RIVEN-AGENT` program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

---

**Figure 2.4**
The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure ??.

```plaintext
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
```

**Figure 2.6**
A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

```plaintext
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
```

---

**Function Simple-Reflex-Agent**

This function implements a simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.
In case the agent’s history in addition to the actual percept is required to decide on the next action, it must be represented in a suitable form.
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
Model-based, Goal-based Agents

- Often, percepts alone are insufficient to decide what to do.
- This is because the correct action depends on the given explicit goals (e.g., go towards X).
- The model-based, goal-based agents use an explicit representation of goals and consider them for the choice of actions.
Model-based, Goal-based Agents

Agent

Environment

Sensors

State

What the world is like now

How the world evolves

What my actions do

What it will be like if I do action A

Goals

What action I should do now

Actuators
Usually, there are several possible actions that can be taken in a given situation.

In such cases, the utility of the next achieved state can come into consideration to arrive at a decision.

A utility function maps a state (or a sequence of states) onto a real number.

The agent can also use these numbers to weigh the importance of competing goals.
Model-based, Utility-based Agents

- **Agent**
  - State
  - How the world evolves
  - What my actions do
  - Utility

- **Environment**
  - Sensors
  - What the world is like now
  - What it will be like if I do action A
  - How happy I will be in such a state
  - What action I should do now

- **Actuators**
Learning Agents

- Learning agents can become more competent over time.
- They can start with an initially empty knowledge base.
- They can operate in initially unknown environments.
Components of Learning Agents

- **learning element** (responsible for making improvements)
- **performance element** (has to select external actions)
- **critic** (determines the performance of the agent)
- **problem generator** (suggests actions that will lead to informative experiences)
Learning Agents

Performance standard
Agent
Environment
Sensors
Performance
element
changes
knowledge
learning
  goals
Problem
generator 
feedback
  Learning  
element
Critic
Actuators

(University of Freiburg) Foundations of AI April 27, 2012 20 / 23
The Environment of Rational Agents

- **Accessible vs. inaccessible (fully observable vs. partially observable)**
  Are the relevant aspects of the environment accessible to the sensors?

- **Deterministic vs. stochastic**
  Is the next state of the environment completely determined by the current state and the selected action? If only actions of other agents are nondeterministic, the environment is called **strategic**.

- **Episodic vs. sequential**
  Can the quality of an action be evaluated within an episode (perception + action), or are future developments decisive for the evaluation of quality?

- **Static vs. dynamic**
  Can the environment change while the agent is deliberating? If the environment does not change but if the agent’s performance score changes as time passes by the environment is denoted as **semi-dynamic**.

- **Discrete vs. continuous**
  Is the environment discrete (chess) or continuous (a robot moving in a room)?

- **Single agent vs. multi-agent**
  Which entities have to be regarded as agents? There are **competitive** and **cooperative** scenarios.
## Examples of Environments

<table>
<thead>
<tr>
<th>Task</th>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
<th>Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossword puzzle</td>
<td>fully</td>
<td>deterministic</td>
<td>sequential</td>
<td>static</td>
<td>discrete</td>
<td>single</td>
</tr>
<tr>
<td>Chess with a clock</td>
<td>fully</td>
<td>strategic</td>
<td>sequential</td>
<td>semi</td>
<td>discrete</td>
<td>multi</td>
</tr>
<tr>
<td>Poker</td>
<td>partially</td>
<td>stochastic</td>
<td>sequential</td>
<td>static</td>
<td>discrete</td>
<td>multi</td>
</tr>
<tr>
<td>Backgammon</td>
<td>fully</td>
<td>stochastic</td>
<td>sequential</td>
<td>static</td>
<td>discrete</td>
<td>multi</td>
</tr>
<tr>
<td>Taxi driving</td>
<td>partially</td>
<td>stochastic</td>
<td>sequential</td>
<td>dynamic</td>
<td>continuous</td>
<td>multi</td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>partially</td>
<td>stochastic</td>
<td>sequential</td>
<td>dynamic</td>
<td>continuous</td>
<td>single</td>
</tr>
<tr>
<td>Image analysis</td>
<td>fully</td>
<td>deterministic</td>
<td>episodic</td>
<td>semi</td>
<td>continuous</td>
<td>single</td>
</tr>
<tr>
<td>Part-picking robot</td>
<td>partially</td>
<td>stochastic</td>
<td>episodic</td>
<td>dynamic</td>
<td>continuous</td>
<td>single</td>
</tr>
<tr>
<td>Refinery controller</td>
<td>partially</td>
<td>stochastic</td>
<td>sequential</td>
<td>dynamic</td>
<td>continuous</td>
<td>single</td>
</tr>
<tr>
<td>Interactive English tutor</td>
<td>partially</td>
<td>stochastic</td>
<td>sequential</td>
<td>dynamic</td>
<td>discrete</td>
<td>multi</td>
</tr>
</tbody>
</table>

Whether an environment has a certain property also depends on the conception of the designer.
An agent is something that perceives and acts. It consists of an architecture and an agent program.

An ideal rational agent always takes the action that maximizes its performance given the percept sequence and its knowledge of the environment.

An agent program maps from a percept to an action.

There are a variety of designs
- Reflex agents respond immediately to percepts.
- Goal-based agents work towards goals.
- Utility-based agents try to maximize their reward.
- Learning agents improve their behavior over time.

Some environments are more demanding than others.

Environments that are partially observable, nondeterministic, strategic, dynamic, and continuous and multi-agent are the most challenging.