Foundations of Artificial Intelligence

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

Joschka Boedecker and Wolfram Burgard and
Frank Hutter and Bernhard Nebel and
Michael Tangermann

Albert-Ludwigs-Universität Freiburg

April 26, 2019
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
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, ...
Lecture Overview

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
Rational Agents

... do the “right thing”!

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

Autonomous vacuum cleaner example:
- Square meters 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.
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

Percept Sequence $\times$ World Knowledge $\rightarrow$ 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, 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>
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
Realization of the ideal mapping through an

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

\[ \text{Agent} = \text{Architecture} + \text{Program} \]
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
The Simplest Design: Table-Driven Agents

function TABLE-DRIVEN-AGENT(percept) returns an action

persistent: percept, a sequence, initially empty
table, a table of actions, indexed by percept sequences, initially fully specified

append percept to the end of percept
action ← LOOKUP(percepts, table)
return action

Problems:

- The table can become very large
- and it usually takes a very long time for the designer to specify it (or to learn it)
- ... practically impossible
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.
Since storage space required for perceptions is too large, direct interpretation of perceptions is impractical. Therefore, storage space for perceptions is limited, and an Immediate Interpretive Reflex Agent (T-DRAgent) is used to handle perceptions.

**Function T-DRAgent**:

```plaintext
function T-DRAgent(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
```

**Function T-DRAgent** is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

**Function REFLEX-ACENT**:

```plaintext
function REFLEX-ACENT([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
```

**Function REFLEX-ACENT** is the agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure ??.

**Function SIMPLE-REFLEX-AGENT**:

```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** is 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.
A Model-based Reflex Agent

```latex
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

- State
- How the world evolves
- What my actions do

Environment

- Sensors

Goals

- What it will be like if I do action A

- What the world is like now

- What action I should do now

Actuators

- What the world is like now
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.
Agent

Environment

Model-based, Utility-based Agents

- State
  - What the world is like now
  - How the world evolves
  - What my actions do

- Utility
  - How happy I will be in such a state
  - What it will be like if I do action A

- Actuators

- Sensors
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

Diagram:

- Performance standard
- Critic
- Sensors
- Learning element
- Changes
- Knowledge
- Problem generator
- Learning goals
- Environment
- Actuators
- Feedback

University of Freiburg
Foundations of AI
April 26, 2019
Lecture Overview

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
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.