What is an Autonomous Agent?

Interacts with its environment

1. can sense its environment
2. make decisions, and
3. take action

Sensors: eyes, sonar, ...

Environment

Agent

Actions

Effectors: hands, motors, ...

Agents

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

Software Agents - Softbots

- “Live” inside computers and networks
- Percepts: text, images, bits
- Actions: display information, send messages, ...
- Examples: simulated pilots, Electric Elves, chat room ‘bots’, MS “paperclip”, ...

Software - Softbots

(Physical) Types of Autonomous Agents

Biological

Agent

Software - Softbots

Hardware - Robots

Artificial Life

Task specific

Entertainment

Sensors: networks, keyboards, ...

Environment

Agent

Actions

Effectors: networks, monitors...
Robotic Agents

- Physical hardware interacts with the environment
- Percepts: images, sound, pressure, acceleration, …
- Actions: movement, sound, light, …

Agents and environments

- The agent function maps from percept histories to actions: $[f: P^* \rightarrow \alpha]$.
- The agent program runs on the physical architecture to produce $f$.
- $agent = architecture + program$

Tiny 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}
- \url{http://aima.cs.berkeley.edu/lisp/doc/overview.html}

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.

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 Description

- **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:

1. Performance measure
2. Environment
3. Actuators
4. Sensors

Taxi Example - PEAS

- First specify the setting for 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

Medical Example: 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)

Example Robot : 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
English Tutor Example: 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

Environment Characteristics (I)

- **Fully observable** (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.
- **Deterministic** (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)
- **Episodic** (vs. sequential): The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.

Environment Characteristics (II)

- **Static** (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is semidynamic if the environment itself does not change with the passage of time but the agent's performance score does)
- **Discrete** (vs. continuous): A limited number of distinct, clearly defined percepts and actions.
- **Single agent** (vs. multiagent): An agent operating by itself in an environment.

Environment Type Examples

<table>
<thead>
<tr>
<th></th>
<th>Chess with a clock</th>
<th>Chess without Taxi driving a clock</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fully observable</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Deterministic</strong></td>
<td>Strategic</td>
<td>Strategic</td>
</tr>
<tr>
<td><strong>Episodic</strong></td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Static</strong></td>
<td>Semi</td>
<td>Strategic</td>
</tr>
<tr>
<td><strong>Discrete</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Single agent</strong></td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Quiz

<table>
<thead>
<tr>
<th></th>
<th>Solitaire</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable??</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic??</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Episodic??</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static??</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Discrete??</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Single-agent??</td>
<td>Yes</td>
<td>No</td>
<td>Yes (except auctions)</td>
<td>No</td>
</tr>
</tbody>
</table>
Agent functions and programs

- An agent is completely specified by the \textbf{agent function} mapping percept sequences to actions
- One agent function (or a small equivalence class) is \textbf{rational}
- Aim: find a way to implement the rational agent function concisely

Table-lookup agent

\begin{itemize}
\item Drawbacks:
  \begin{itemize}
  \item Huge table
  \item Take a long time to build the table
  \item No autonomy
  \item Even with learning, need a long time to learn the table entries
  \end{itemize}
\end{itemize}

A vacuum-cleaner agent

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
Percept sequence & Action \\
\hline
[A, Clean] & Right \\
A, Dirty & Suck \\
B, Clean & Left \\
B, Dirty & Suck \\
A, Clean, [A, Clean] & Right \\
[A, Clean], [A, Dirty] & Suck \\
\hline
\end{tabular}
\caption{Table for a vacuum-cleaner agent}
\end{table}

\begin{verbatim}
function ReflexVacuum-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
\end{verbatim}

Agent program for a vacuum-cleaner agent

\begin{verbatim}
\input{algorithms/reflex-vacuum-agent-algorithm}
\end{verbatim}

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

\begin{figure}
\centering
\includegraphics{simple-reflex-agent}
\caption{Simple reflex agent diagram}
\end{figure}
Artificial Intelligence

Simple reflex agents

```lisp
(defun 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)
```

```lisp
(defun make-reflex-vacuum-agent-program ()
  (let ((percept)
        (first-percept)
        (second-percept)
        (status (eq status 'dirty) 'Suck)
        (eq location 'A) 'Right)
        (eq location 'B) 'Left))))
```

Model-based reflex agents

```lisp
(defun reflex-vacuum-agent-with-state-program ()
  (let ((percept)
        (first-percept)
        (second-percept)
        (last-A inf)
        (last-B inf)
        (status (eq status 'dirty) 'Suck)
        (eq location 'A) (eq location 'B) (eq location 'C)
        (eq location 'D) (eq location 'E) (eq location 'F)
        (eq location 'G) (eq location 'H) (eq location 'I)
        (eq location 'J) (eq location 'K) (eq location 'L)
        (eq location 'M) (eq location 'N) (eq location 'O)
        (eq location 'P) (eq location 'Q) (eq location 'R)
        (eq location 'S) (eq location 'T) (eq location 'U)
        (eq location 'V) (eq location 'W) (eq location 'X)
        (eq location 'Y) (eq location 'Z))
  (cond
    ((eq status 'dirty) 'Suck)
    ((eq location 'A) (eq location 'B) (eq location 'C)
     (eq location 'D) (eq location 'E) (eq location 'F)
     (eq location 'G) (eq location 'H) (eq location 'I)
     (eq location 'J) (eq location 'K) (eq location 'L)
     (eq location 'M) (eq location 'N) (eq location 'O)
     (eq location 'P) (eq location 'Q) (eq location 'R)
     (eq location 'S) (eq location 'T) (eq location 'U)
     (eq location 'V) (eq location 'W) (eq location 'X)
     (eq location 'Y) (eq location 'Z)))
```

Goal-based agents

Utility-based agents

Learning agents
Summary

Agents interact with environments through actuators and sensors.

The agent function describes what the agent does in all circumstances.

The performance measure evaluates the environment sequence.

A perfectly rational agent maximizes expected performance.

Agent programs implement (some) agent functions.

PEAS descriptions define task environments.

Environments are categorized along several dimensions:
  - observable?
  - deterministic?
  - episodic?
  - static?
  - discrete?
  - single-agent?

Several basic agent architectures exist:
  - reflex
  - reflex with state
  - goal-based
  - utility-based