Intelligent Agents

Readings: Chapter 2 of Russell & Norvig.
AI is pretty hard stuff

I went to the grocery store, I saw the milk on the shelf and I bought it.

What did I buy?

- The milk?
- The shelf?
- The store?

An awful lot of knowledge of the world is needed to answer simple questions like this one.
Agents and Environments

An agent is a system that perceives its environment through sensors and acts upon that environment through effectors.

Agents include humans, robots, softbots, thermostats, etc.
Agents as Mappings

An agent can be seen as a mapping between percept sequences and actions.

\[ \text{Agent} : \text{Percept}^* \rightarrow \text{Action}^* \]

The less an agent relies on its built-in knowledge, as opposed to the current percept sequence, the more autonomous it is.

A rational agent is an agent whose acts try to maximize some performance measure.
Vacuum-cleaner world

Percepts: location and contents, 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, Clean])</td>
<td>Right</td>
</tr>
<tr>
<td>([A, Dirty])</td>
<td>Suck</td>
</tr>
<tr>
<td>([B, Clean])</td>
<td>Left</td>
</tr>
<tr>
<td>([B, Dirty])</td>
<td>Suck</td>
</tr>
<tr>
<td>([A, Clean], [A, Clean])</td>
<td>Right</td>
</tr>
<tr>
<td>([A, Clean], [A, Dirty])</td>
<td>Suck</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**function** \[ \text{REFLEX-VACUUM-AGENT( [location, status] )} \]

**returns**

```
action

if status = \text{Dirty} \text{ then return } Suck
else if location = A \text{ then return } Right
else if location = B \text{ then return } Left
```
## More Examples of Artificial Agents

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Percepts</th>
<th>Actions</th>
<th>Goals</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medical diagnosis system</td>
<td>Symptoms, findings, patient’s answers</td>
<td>Questions, tests, treatments</td>
<td>Healthy patient, minimize costs</td>
<td>Patient, hospital</td>
</tr>
<tr>
<td>Satellite image analysis system</td>
<td>Pixels of varying intensity, color</td>
<td>Print a categorization of scene</td>
<td>Correct categorization</td>
<td>Images from orbiting satellite</td>
</tr>
<tr>
<td>Part-picking robot</td>
<td>Pixels of varying intensity</td>
<td>Pick up parts and sort into bins</td>
<td>Place parts in correct bins</td>
<td>Conveyor belt with parts</td>
</tr>
<tr>
<td>Refinery controller</td>
<td>Temperature, pressure readings</td>
<td>Open, close valves; adjust temperature</td>
<td>Maximize purity, yield, safety</td>
<td>Refinery</td>
</tr>
<tr>
<td>Interactive English tutor</td>
<td>Typed words</td>
<td>Print exercises, suggestions, corrections</td>
<td>Maximize student’s score on test</td>
<td>Set of students</td>
</tr>
</tbody>
</table>
Rationality

- What is the right function?
- Can it be implemented in a small agent program?
- Fixed performance measure evaluates the environment sequence
  - one point per square cleaned up in time $T$?
  - one point per clean square per time step, minus one per move?
  - penalize for $> k$ dirty squares?
- Rational $\neq$ omniscient
- Rational $\neq$ successful
- Rational $\implies$ exploration, learning, autonomy
Rational Agents

The *rationality* of an agent depends on

- the **performance measure** defining the agent’s degree of success

- the **percept sequence**, the sequence of all the things perceived by the agent

- the agent’s **knowledge** of the environment

- the **actions** that the agent can perform

For each possible percept sequence, an *ideal* rational agent does whatever possible to maximize its performance, based on the percept sequence and its built-in knowledge.
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
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
- Environment
- Actuators
- Sensors
Environment Types

With respect to an agent, an environment may, or may not, be:

- **accessible**: the agent’s sensors detect all aspects relevant to the choice of action;
- **deterministic**: the next state is completely determined by the current state and the actions selected by the agent;
- **episodic**: the agent’s experience is divided into “episodes”; the quality of the agent’s actions does not depend on previous episodes;
- **static**: it does not change while the agent is deliberating;
- **discrete**: there are a limited number of distinct, clearly defined percepts and actions.
## Environment Types

<table>
<thead>
<tr>
<th>Observable??</th>
<th>Deterministic??</th>
<th>Episodic??</th>
<th>Static??</th>
<th>Discrete??</th>
<th>Single-agent??</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solitaire</td>
<td>Backgammon</td>
<td>E-shopping</td>
<td>Taxi</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Environment types

<table>
<thead>
<tr>
<th>Observable??</th>
<th>Deterministic??</th>
<th>Episodic??</th>
<th>Static??</th>
<th>Discrete??</th>
<th>Single-agent??</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solitaire</th>
<th>Backgammon</th>
<th>E-shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
## Environment types

<table>
<thead>
<tr>
<th></th>
<th>Solitaire</th>
<th>Backgammon</th>
<th>E-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>Partly</td>
<td>No</td>
</tr>
<tr>
<td>Episodic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-agent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Environment types

<table>
<thead>
<tr>
<th></th>
<th>Solitaire</th>
<th>Backgammon</th>
<th>E-shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Observable</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Deterministic</strong></td>
<td>Yes</td>
<td>No</td>
<td>Partly</td>
<td>No</td>
</tr>
<tr>
<td><strong>Episodic</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Static</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Discrete</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Single-agent</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Environment types

<table>
<thead>
<tr>
<th></th>
<th>Solitaire</th>
<th>Backgammon</th>
<th>E-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>Partly</td>
<td>No</td>
</tr>
<tr>
<td>Episodic??</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static??</td>
<td>Yes</td>
<td>Semi</td>
<td>Semi</td>
<td>No</td>
</tr>
<tr>
<td>Discrete??</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-agent??</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Environment types

<table>
<thead>
<tr>
<th></th>
<th>Solitaire</th>
<th>Backgammon</th>
<th>E-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>Partly</td>
<td>No</td>
</tr>
<tr>
<td>Episodic</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static</td>
<td>Yes</td>
<td>Semi</td>
<td>Semi</td>
<td>No</td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Single-agent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Environment types

<table>
<thead>
<tr>
<th></th>
<th>Solitaire</th>
<th>Backgammon</th>
<th>E-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>Partly</td>
<td>No</td>
</tr>
<tr>
<td>Episodic</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static</td>
<td>Yes</td>
<td>Semi</td>
<td>Semi</td>
<td>No</td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Single-agent</td>
<td>Yes</td>
<td>No</td>
<td>Yes/No</td>
<td>No</td>
</tr>
</tbody>
</table>
## Environment types: Vacuum-Cleaner

Percepts: location and contents, e.g., \([A, \text{Dirty}]\)

Actions: \(\text{Left}, \text{Right}, \text{Suck}, \text{NoOp}\)

<table>
<thead>
<tr>
<th></th>
<th>Real World</th>
<th>Simplified World</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable??</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deterministic??</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Episodic??</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static??</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete??</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-agent??</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Environment types

The environment type largely determines the agent design

The real world is (of course)

- partially *observable*,
- stochastic (instead of *deterministic*),
- sequential (instead of *episodic*),
- dynamic (instead of *static*),
- continuous (instead of discrete),
- multi-agents (instead of *single-agent*).
Agent Programs

Since an agent is just a mapping from percepts to actions, we can design a program to implement this mapping.

An agent program could be as simple as a table lookup. However:

- that might be impossible
- there might be a much more efficient solution
- the agent would have no autonomy

\[^a\] A chess playing agent, for instance, would need \(35^{100}\) table entries.
Different Types of Agents

Agents programs can be divided in the following classes, with increasing level of sophistication:

- Simple reflex agents
- Goal-based agents
- Utility-based agents
- Learning agents
A Reflex Taxi-Driver Agent

- We cannot implement it as a table-lookup: the percepts are too complex.

- But we can abstract some portions of the table by coding common input/output associations.

- We do this with a list of condition/action rules:
  - if car-in-front-is-braking then brake
  - if light-becomes-green then move-forward
  - if intersection-has-stop-sign then stop
Simple Reflex Agents

Reflex agents can be implemented very efficiently.

However, they have limited applicability.
Reflex Taxi-Driver Agent with State

- Often, the agent must remember some of its percepts to take an action.
  
  *Ex: car in front signals it is turning left.*

- It must also remember which actions it has taken.
  
  *Ex: loaded/unloaded passenger.*

- In jargon, it must have internal state.
Reflex Agents with Internal State

Agent

Environment

- Sensors
- State
- How the world evolves
- What my actions do
- Condition–action rules
- What the world is like now
- What action I should do now
- Actuators
Reflex Taxi-Driver Agent with State

To update its state the agent needs two kinds of knowledge:

1. how the world evolves independently from the agent;
   Ex: an overtaking car gets closer with time.

2. how the world is affected by the agent’s actions.
   Ex: if I turn left, what was to my right is now behind me.
A Goal-based Taxi-Driver Agent

Knowing about the world is not always enough to decide what to do.

*Ex: what direction do I take at an intersection?*

The agent needs **goal** information.

*Ex: passenger’s destination*

Combining goal information with the knowledge of its actions, the agent can choose those actions that will achieve the goal.

A new kind of decision-making is required ("what-if reasoning").

**Search** and **Planning** are devoted to find action sequences that achieve an agent’s goal.
Goal-based Agents are much more flexible in
- responding to a changing environment;
- accepting different goals.
Utility-based Taxi-Driver Agent

- There may be many ways to get to a destination but some may be better than others.
  
  *Ex: this way is faster/cheaper/more comfortable*... 

- A particular configuration of the world, a *world state*, can be assigned a *utility* (the quality of being useful) value for the agent.

- A sequence of actions is preferred if it leads to a goal state with higher utility value.

- A *utility function* helps the agent’s decision-making in case of
  
  1. conflicting goals (by helping find a trade-off).
    
    *Ex: minimize trip time and also fuel consumption.*
  
  2. several possible goals, none of which is achievable with certainty.
Utility-based Agents

Agent

- State
- What 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

Agent

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