INTELLIGENT AGENTS

CHAPTER 2
Reminders

CLASS WILL MOVE to MonWed 4-5.30 in 10 Evans starting 9/9

Assignment 0 (lisp refresher) due 9/4

Lisp/emacs/ALMA tutorial: Wed Aug 28th, 4-6pm, 273 Soda

Class account forms
Outline

◊ Agents and environments
◊ Rationality
◊ PEAS (Performance measure, Environment, Actuators, Sensors)
◊ Environment types
◊ Agent types
Agents include humans, robots, softbots, thermostats, etc.

The agent function maps from percept histories to actions:

\[ f : \mathcal{P}^* \rightarrow \mathcal{A} \]

The agent program runs on the physical architecture to produce \( f \)
Vacuum-cleaner world

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

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>Right</td>
</tr>
<tr>
<td>([A, \text{Dirty}])</td>
<td>Suck</td>
</tr>
<tr>
<td>([B, \text{Clean}])</td>
<td>Left</td>
</tr>
<tr>
<td>([B, \text{Dirty}])</td>
<td>Suck</td>
</tr>
<tr>
<td>([A, \text{Clean}], [A, \text{Clean}])</td>
<td>Right</td>
</tr>
<tr>
<td>([A, \text{Clean}], [A, \text{Dirty}])</td>
<td>Suck</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

```python
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
```

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

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?

A rational agent chooses whichever action maximizes the expected value of
the performance measure given the percept sequence to date

Rational $\neq$ omniscient
Rational $\neq$ clairvoyant
Rational $\neq$ successful

Rational $\Rightarrow$ exploration, learning, autonomy
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??**
PEAS

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??

Environment??

Actuators??

Sensors??
## 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>Internet shopping</td>
<td>Taxi</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---
<table>
<thead>
<tr>
<th>Environment types</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></td>
<td></td>
<td></td>
<td></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>Observable??</th>
<th>Solitaire</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic??</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Episodic??</td>
<td>Yes</td>
<td>No</td>
<td>Partly</td>
<td>No</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>Internet 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>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>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>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>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>Observability</th>
<th>Solitaire</th>
<th>Backgammon</th>
<th>Internet 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>Yes</td>
<td>Semi</td>
<td>Semi</td>
<td>No</td>
</tr>
<tr>
<td><strong>Discrete</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Single-agent</strong></td>
<td>Yes</td>
<td>No</td>
<td>No (except auctions)</td>
<td>No</td>
</tr>
</tbody>
</table>

The environment type largely determines the agent design

The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent
Agent types

Four basic types in order of increasing generality:
- simple reflex agents
- reflex agents with state
- goal-based agents
- utility-based agents

All these can be turned into learning agents
Simple reflex agents

Agent

Sensors

What the world is like now

Condition–action rules

What action I should do now

Actuators

Environment
Reflex agents with state

Agent

Environment

State

How the world evolves

What my actions do

Condition–action rules

What the world is like now

Sensors

What I should do now

Actuators
Goal-based agents

Agent

- State
- How the world evolves
- What my actions do
- Goals
- What the world is like now
- What it will be like if I do action A
- What action I should do now

Environment

Sensors

Actuators
Utility-based agents

Agent

Environment

State

How the world evolves

What my actions do

Utility

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

Sensors
Learning agents

Performance standard

Critic

Sensors

feedback

changes

knowledge

Learning element

Performance element

learning goals

Problem generator

Agent

Environment

Actuators
The code for each topic is divided into four directories:
- agents: code defining agent types and programs
- algorithms: code for the methods used by the agent programs
- environments: code defining environment types, simulations
- domains: problem types and instances for input to algorithms

(Often run algorithms on domains rather than agents in environments.)

(setq joe (make-agent :name 'joe :body (make-agent-body)
                      :program (make-dumb-agent-program)))

(defun make-dumb-agent-program ()
  (let ((memory nil))
    #'(lambda (percept)
        (push percept memory)
        'no-op)))