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., \([A, Dirty]\)

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>

What is the right function?
Can it be implemented in a small agent program?
function REFLEX-VACUUM-AGENT [\textit{location, status}] 
returns an action 
if status = \textit{Dirty} then return \textit{Suck}
else if location = A then return \textit{Right}
else if location = B then return \textit{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?
- 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 $\implies$ 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:**
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>Internet</td>
<td></td>
<td>Solitaire</td>
<td>Backgammon</td>
<td>shopping</td>
<td>Taxi</td>
</tr>
</tbody>
</table>

- **Fully Observable**: access to the complete (relevant) state of the world
- **Partially Observable**: missing information
### Environment types

<table>
<thead>
<tr>
<th>Observable??</th>
<th>Deterministic??</th>
<th>Episodic??</th>
<th>Static??</th>
<th>Discrete??</th>
<th>Single-agent??</th>
<th>Solitaire</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes (?)</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes (??)</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

**Deterministic**: the next state is completely determined by the current state and the action

**Stochastic**: Changes not known

**Strategic**: Deterministic except for the actions of the other agents
### 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>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>

*Episodic*: task divided into atomic episodes  
*Sequential*: Current decision may affect all future decisions
### 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>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Discrete??</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Single-agent??</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

**Static**: the world does not change while the agent is thinking

**Dynamic**: changes

**Semidynamic**: does not change but the performance is affected as time passes
## 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>no</td>
<td>no</td>
</tr>
<tr>
<td>Discrete??</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Single-agent??</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

*Discrete*: time, percepts, and actions are discrete

*Continuous*: time, percepts, and actions are continuous over time
## 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>yes (?)</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>no</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>

**Single-agent**: one agent

**Multi-agent**: competitive or cooperating agents
### 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>no</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 (?)</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

Environment

Sensors

What the world is like now

Condition-action rules

What action I should do now

Actuators
Reflex agents with state

Agent

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

Condition-action rules

Environment

- Sensors
- What the world is like now
- What action I should do now

Actuators
Goal-based agents

Agent

State

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

Sensors

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
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)))