**Intelligent Agents**

AIMA Chapter 2, 2nd Ed. (after Russell and Norvig)

Agents Interact with Environments

Must first specify the setting for intelligent agent design

An agent perceives its environment through sensors and acts upon it through actuators

Example Sensors and Actuators

- Humans
- Robots
- Softbots
Example Sensors and Actuators

**Humans**? eyes and ears / hands and legs

**Robot**? cameras / motors

**Softbot**? keystrokes / displays

---

Agents and Environments (cont.)

Mathematically, an *agent function* maps any percept sequence to an action (and thus describes behavior)

- percepts: agent’s perceptual inputs at any instance
- percept sequence: complete history
- action: an agent’s action choice at any instant can depend on the entire percept sequence

Problematic from an implementation perspective (why?), so need *agent programs*

---

Examples (cont.)

Consider the task of designing an automated taxi:

**Percepts**?

**Actions**?

**Environment**?

---

Examples (cont.)

Consider the task of designing an automated taxi:

**Percepts**? video, accelerometers, gauges, engine sensors, keyboard, GPS, ...

**Actions**? steer, accelerate, brake, horn, speak/display, ...

**Environment**? US urban streets, freeways, traffic, pedestrians, weather, customers, ...


**Another Example: Vacuum World**

- **Percepts??**
  - location, dirtiness
- **Actions??**
  - suck, left, right, no-op
- **Environment??**
  - grid, walls/obstacles, dirt distribution and creation, agent body (movement actions work unless bump into wall, suck actions put dirt into agent body (or not))

---

**Simple Agent Function for Vacuum World**

Partial tabulation of this simple agent function

<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>
</tbody>
</table>

How can we define different vacuum world agents?

What is the obvious question for AI?

---

**Agent Program**

Agent function: If the current square is dirty, then suck dirt; otherwise, move to the other square.

```
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
```

**Good Behavior: Rationality**

A rational agent is one that does “the right thing”, e.g., every entry in the action function table is filled out correctly

- the right action is the one that will cause the agent to be most successful
- therefore, we need to be able to measure success
- a performance measure embodies the criterion for success of an agent’s behavior

**Performance Measures**

Performance measure: an objective, numerical value for any environment history

What are reasonable performance measures for the vacuum world?

- the amount of dirt cleaned up in an hour – 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?
- having a clean floor
- generally better to measure what you want in the environment, rather than how you think the agent should behave
- difficult to come up with measures (sustained mediocrity vs. highs and lows)

**Rationality**

Rationality depends on

- the performance measure defining the success criterion
- the agent’s prior knowledge of the environment
- the actions that the agent can perform
- the agent’s percept sequence to date

Rational action: whichever action maximizes the expected value of the performance measure given the percept sequence to date and built-in knowledge

Rational agent: for each possible percept sequence, selects an action that is expected to maximize its performance measure

Rational ≠ omniscient
Rational ≠ clairvoyant
Rational ≠ successful
Omniscience, Learning, and Autonomy

Rational \( \neq \) omniscient

- airplane flattens person crossing street example
- rationality maximizes expected performance, depending on knowledge to date; perfection maximizes actual performance
- crossing without looking is not rational because lacks information gathering (doing actions to modify future percepts, exploration)

Rational agents should also

- learn from percepts (to augment or modify prior knowledge)
- learn to be autonomous (rely on percepts rather than prior – often partial and/or incorrect – knowledge)

Rational \( \Rightarrow \) exploration, learning, autonomy

Specifying the Task Environment: PEAS

Task Environments: “problems” to which “agents” are solutions

We thus need to specify the problem before we develop the solution

Example: PEAS Specification for an Automated Taxi Driver Agent

- Performance Measures: correct destination, safe, fast, legal, comfortable, profitable, 
- Environment: roads, traffic, pedestrians, customers, 
- Actuators: steering, accelerator, brake, horn, 
- Sensors: camera, sonar, speedometer, GPS, 

More PEAS Examples

Text-based Conversational Tutor

- performance: maximize test score
- environment: students, testing agency
- actuators: display exercise, suggestions, corrections
- sensors: keyboard entry

What about a Speech-based Conversational Tutor?

See Figure 2.5 for more examples

NOTE: toy \( \neq \) artificial environment

Environment Dimensions

Fully versus Partially Observable

- fully is with respect to observation relevance for action choice (thus depends on performance measure)
- often partial due to noise and incompleteness

Deterministic versus Stochastic

- deterministic if next environment state is completely determined by the current state and action choice

Episodic versus Sequential

- episodic: independent episodes (current percept, then perform a single action, e.g., assembly line)
- sequential: short term actions can have long term consequences
Static versus Dynamic

- dynamic: environment can change during thought
- semidynamic: environment doesn’t change with time but performance score does

Discrete versus Continuous

- can be applied to environment state, time, percepts and actions

Single versus Multi Agent

- other agents if their behavior is maximizing a performance measure based on first agent’s behavior
- multi-agents can be cooperative, competitive (which can impact choice of communication actions and stochastic behavior)

What is the hardest environment?

<table>
<thead>
<tr>
<th>Environment Dimension</th>
<th>Crossword</th>
<th>Backgammon</th>
<th>Tutor</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>No</td>
<td>No</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>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Agents??</td>
<td>Single</td>
<td>Multi</td>
<td>Multi</td>
<td>Multi</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

See also Figure 2.6

Agent Functions

An agent is completely specified by the agent function mapping percept sequences to actions (desirable behavior)

- In principle, one can supply each possible sequence to see what it does. Obviously, a lookup table would usually be immense.
- One agent function (or a small equivalence class) is rational
### Agent Programs

The job of AI is to design the agent program that implements the agent function – concisely

- agent = architecture + program

An agent program takes a single percept as input, keeps internal state:

```
function Table-Driven-Agent(percept) returns an action
    static: percepts, a sequence, initially empty
    table, a table of actions, indexed by percept sequences, initially fully specified
    percepts ← Append(percept, percepts)
    action ← Lookup(percepts, table)
    return action
```

### Agent Types

Four basic types in order of increasing generality:

- simple reflex agents
- model-based reflex agents with state
- goal-based agents
- utility-based agents

All these can be turned into learning agents

### Simple Reflex Agents

Action selection is based on the current percept (assumes fully observable environment)

Condition-action rules (e.g., if car-in-front-is-breaking then initiate-breaking) represent both innate and learned reflexes

See Simple Reflex Agent Figure

### Simple Reflex Agent Programs: Examples

Figures 2.8 (specific to vacuum world) and 2.10 (generalization)

Note that the programs are smaller than the function they implement (Figure 2.3)
Model-Based Reflex Agents with State

State handles partial observability
State is updated with the model (how the world evolves, agent’s actions): interpret-input(percept) replaced with update-state(state,action,percept)
See Reflex+State Agent Figure

Goal-Based Agents

Search and planning deal with tricky, goal-based action (sequence) selection
These agents consider the future (e.g., brake via reasoning, not just reflex)
See Goal-Based Agent Figure

Utility-Based Agents

Goals are just binary
A utility function maps a state onto a real number representing a preference order
Useful for conflicting goals and goal choice
See Utility-Based Agent Figure

Learning Agents

Previously, concerned with methods for action selection in the agent program
Learning is how programs come into being, and improve
Performance element was previously the agent; problem generator is for exploration
See Learning Agent Figure
Summary

Agent: something that perceives and acts in an environment

Agent function: specifies the action taken in response to any percept sequence

Performance measure: evaluates the agent's behavior in an environment

Rational agent: acts to maximize the expected value of the performance measure given the percept sequence to date

Task environment: specification via PEAS, many dimensions (e.g., static or dynamic)

Agent program: implements the agent function

Agent designs: best choice (e.g., simple reflex) depends on environment

Learning agents: improve performance via learning