CS 331: Artificial Intelligence
Intelligent Agents
General Properties of AI Systems

This part is called an **agent**.

**Agent**: anything that perceives its environment through sensors and acts on that environment through actuators
Example: 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>
<tr>
<td>[A, Clean], [A, Clean], [A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Clean], [A, Clean], [A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
Agent-Related Terms

- **Percept sequence**: A complete history of everything the agent has ever perceived. Think of this as the state of the world from the agent’s perspective.

- **Agent function (or Policy)**: Maps percept sequence to action (determines agent behavior)

- **Agent program**: Implements the agent function
Question

What’s the difference between the agent function and the agent program?
Rationality

- Rationality: do the action that causes the agent to be most successful
- How do you define success? Need a performance measure
- E.g. reward agent with one point for each clean square at each time step (could penalize for costs and noise)

Important point: Design performance measures according to what one wants in the environment, not according to how one thinks the agent should behave
Rationality

Rationality depends on 4 things:
1. Performance measure of success
2. Agent’s prior knowledge of environment
3. Actions agent can perform
4. Agent’s percept sequence to date

**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.
Learning

Successful agents split task of computing policy in 3 periods:

1. Initially, designers compute some prior knowledge to include in policy
2. When deciding its next action, agent does some computation
3. Agent learns from experience to modify its behavior

Autonomous agents: Learn from experience to compensate for partial or incorrect prior knowledge
Example: Automated taxi driver

<table>
<thead>
<tr>
<th>Performance Measure</th>
<th>Environment</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe, fast, legal, comfortable trip, maximize profits</td>
<td>Roads, other traffic, pedestrians, customers</td>
<td>Steering, accelerator, brake, signal, horn, display</td>
<td>Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard</td>
</tr>
</tbody>
</table>
## Properties of Environments

<table>
<thead>
<tr>
<th>Property</th>
<th>vs</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully observable: can access complete state of environment at each point in time</td>
<td>vs Partially observable: could be due to noisy, inaccurate or incomplete sensor data</td>
<td><strong>Deterministic</strong>: if next state of the environment completely determined by current state and agent’s action</td>
</tr>
<tr>
<td><strong>Episodic</strong>: agent’s experience divided into independent, atomic episodes in which agent perceives and performs a single action in each episode.</td>
<td>vs <strong>Sequential</strong>: current decision affects all future decisions</td>
<td><strong>Static</strong>: agent doesn’t need to keep sensing while decides what action to take, doesn’t need to worry about time</td>
</tr>
<tr>
<td><strong>Discrete</strong>: (note: discrete/continuous distinction applies to states, time, percepts, or actions)</td>
<td>vs <strong>Continuous</strong></td>
<td><strong>Single agent</strong></td>
</tr>
</tbody>
</table>
## Examples of task environments

<table>
<thead>
<tr>
<th>Task Environment</th>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
<th>Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossword puzzle</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Single</td>
</tr>
<tr>
<td>Chess with a clock</td>
<td>Fully</td>
<td>Strategic</td>
<td>Sequential</td>
<td>Semi</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td>Poker</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td>Backgammon</td>
<td>Fully</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td>Taxi driving</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Multi</td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Multi</td>
</tr>
<tr>
<td>Image analysis</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Episodic</td>
<td>Semi</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td>Part-picking robot</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Episodic</td>
<td>Semi</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td>Refinery controller</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td>Interactive English tutor</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
</tbody>
</table>
In-class Exercise

Develop a PEAS description of the task environment for a movie recommendation agent

<table>
<thead>
<tr>
<th>Performance Measure</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td></td>
</tr>
<tr>
<td>Actuators</td>
<td></td>
</tr>
<tr>
<td>Sensors</td>
<td></td>
</tr>
</tbody>
</table>
Describe the task environment for the movie recommendation agent

<table>
<thead>
<tr>
<th>Fully Observable</th>
<th>Partially Observable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>Stochastic</td>
</tr>
<tr>
<td>Episodic</td>
<td>Sequential</td>
</tr>
<tr>
<td>Static</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Discrete</td>
<td>Continuous</td>
</tr>
<tr>
<td>Single agent</td>
<td>Multi-agent</td>
</tr>
</tbody>
</table>
Agent Programs

- Agent program: implements the policy
- Simplest agent program is a table-driven agent

```
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 specific

    append percept to the end of percepts

    action ← LOOKUP(percepts, table)

    return action
```

This is a BIG table...clearly not feasible!
4 Kinds of Agent Programs

• Simplex reflex agents
• Model-based reflex agents
• Goal-based agents
• Utility-based agents
**Simple Reflex Agent**

- Selects actions using only the current percept
- Works on condition-action rules:
  
  ```
  if condition then action
  ```

```vbnet
function SIMPLE-REFLEX-AGENT(percept) returns an action

  static: rules, a set of condition-action rules

  state ← INTERPRET-INPUT(percept)
  rule ← RULE-MATCH(state, rules)
  action ← RULE-ACTION[rule]

  return action
```
Simple Reflex Agents
Simple Reflex Agents

• Advantages:
  – Easy to implement
  – Uses much less memory than the table-driven agent

• Disadvantages:
  – Will only work correctly if the environment is fully observable
  – Infinite loops
Model-based Reflex Agents

• Maintain some internal state that keeps track of the part of the world it can’t see now
• Needs model (encodes knowledge about how the world works)

```plaintext
function REFLEX-AGENT-WITH-STATE(percept) returns an action

  static: state, a description of the current world state
  rules, a set of condition-action rules
  action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept)
  rule ← RULE-MATCH(state, rules)
  action ← RULE-ACTION[rule]
  return action
```
Model-based Reflex Agents
Goal-based Agents

• Goal information guides agent’s actions (looks to the future)
• Sometimes achieving goal is simple e.g. from a single action
• Other times, goal requires reasoning about long sequences of actions
• Flexible: simply reprogram the agent by changing goals
Goal-based Agents
Utility-based Agents

• What if there are many paths to the goal?
• Utility measures which states are preferable to other states
• Maps state to real number (utility or “happiness”)
Utility-based Agents
Learning Agents
Learning Agents

Think of this as outside the agent since you don’t want it to be changed by the agent.

Maps percepts to actions.
Learning Agents

Critic: Tells learning element how well the agent is doing with respect to the performance standard (because the percepts don’t tell the agent about its success/failure)

Responsible for improving the agent’s behavior with experience

Suggest actions to come up with new and informative experiences
In-class Exercise

• Select a suitable agent design for the movie recommendation agent
What you should know

- What it means to be rational
- Be able to do a PEAS description of a task environment
- Be able to determine the properties of a task environment
- Know which agent program is appropriate for your task