Artificial Intelligence
CS 6364

Professor Dan Moldovan

Section 2
Intelligent Agents
Intelligent Agents

An agent is a thing (e.g. program, or system) that can be viewed as perceiving its environment and acting upon that environment through effectors.

The notion of agent is useful because it provides a tool for analyzing AI systems.

What do we want from agents?
To act rationally.
Intelligent Agents

- Agents:
  - Perceive the environment through sensors.
  - Perform actions.

Intelligence is based on these two: perception and capability of actions. Agents must perform actions to obtain useful information (acquire more percepts).

An agent may be regarded as a mapping of percepts into actions.

- A performance measure is needed to determine how successful an agent is.

An ideal rational agent provides an action that maximizes its performance measure, based on the percepts received and its built-in knowledge.
Autonomous agents

We want to build intelligent agents capable of reacting to the environment on their own. That is to say that we want agents to be able to learn from their experiences and face new situations.

This is different than programming extensively, hoping that the programmer foresaw all possible situations.

An agent is autonomous when its behavior is determined by its own experience. (autonomous from other agents or humans)

Example: a clock adjusting to time zones.
Intelligent Agents vs Other Software

- Agents are autonomous
- Agents contain some level of intelligence
- Agents react to environments, and can sometimes take proactive actions
- Agents may have social ability; i.e., communicate with user and other agents
- Agents may cooperate with other agents
- Agents may migrate from system to system
Structure of Intelligent Agents

Agents are built with programs running on some hardware. AI systems may be general-purpose computers, or special-purpose computers.

Agent programs are very complex, because intelligence is complex. Simple look up tables mapping percepts into actions will not work.

Example: A chess player would require $35^{100}$ entries.
### Example: Automated Taxi Driver

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Performance Measure</th>
<th>Environment</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxi driver</td>
<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>
## Agent Types and their Descriptions

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Performance Measure</th>
<th>Environment</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medical diagnosis system</td>
<td>Healthy patient, minimize costs, lawsuits</td>
<td>Patient, hospital, staff</td>
<td>Display questions, tests, diagnoses, treatments, referrals</td>
<td>Keyboard entry of symptoms, findings, patient’s answers</td>
</tr>
<tr>
<td>Satellite image analysis system</td>
<td>Correct image categorization</td>
<td>Downlink from orbiting satellite</td>
<td>Display categorization of scene</td>
<td>Color pixel arrays</td>
</tr>
<tr>
<td>Part-picking robot</td>
<td>Percentage of parts in correct bins</td>
<td>Conveyor belt with parts; bins</td>
<td>Jointed arm and hand</td>
<td>Camera, joint angle sensors</td>
</tr>
<tr>
<td>Refinery controller</td>
<td>Maximize purity, yield, safety</td>
<td>Refinery, operators</td>
<td>Valves, pumps, heaters, displays</td>
<td>Temperature, pressure, chemical sensors</td>
</tr>
<tr>
<td>Interactive English tutor</td>
<td>Maximize student’s score on test</td>
<td>Set of students, testing agency</td>
<td>Display exercises, suggestions, corrections</td>
<td>Keyboard entry</td>
</tr>
</tbody>
</table>
Properties of environments

1. **Accessible vs inaccessible**
   Accessible when the sensors detect all aspects that are relevant to the choice of action.

2. **Deterministic vs nondeterministic**
   When the next state of environment is completely determined from the current state and the actions selected by agents.

3. **Episodic vs nonepisodic**
   Episodic environment is when agent’s experience is divided into “episodes”. Agent perceives an episode then acts, and is done with it. Subsequent episodes do not depend on the episode itself.

4. **Static vs dynamic**
   Dynamic is when environment changes while agent deliberates and acts.

5. **Discrete vs continuous**
   Descrete is when there is a limited number of distinct, clearly defined percepts and actions.

*Note: The hardest are: inaccessible, nonepisodic, dynamic and continuous.*
## Properties of environments: examples

<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 Chess with a clock</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Single</td>
</tr>
<tr>
<td></td>
<td>Fully</td>
<td>Strategic</td>
<td>Sequential</td>
<td>Semi</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td>Poker Backgammon</td>
<td>Partially</td>
<td>Strategic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td></td>
<td>Fully</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Static</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
<tr>
<td>Taxi driving Medical diagnosis</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Multi</td>
</tr>
<tr>
<td></td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td>Image-analysis Part-picking robot</td>
<td>Fully</td>
<td>Deterministic</td>
<td>Episodic</td>
<td>Semi</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td></td>
<td>Partially</td>
<td>Stochastic</td>
<td>Episodic</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td>Refinery controller Interactive English tutor</td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Continuous</td>
<td>Single</td>
</tr>
<tr>
<td></td>
<td>Partially</td>
<td>Stochastic</td>
<td>Sequential</td>
<td>Dynamic</td>
<td>Discrete</td>
<td>Multi</td>
</tr>
</tbody>
</table>
The Structure of Intelligent Agents

The question facing the designer is: How to structure (architect) such an agent?

Four types of agents (in increasing complexity order)
- Simple reflex agents
- Agents that keep track of the world
- Goal-based agents
- Utility-based agents
Simple reflex agents

- These are implemented with condition-action rules.
  
  if condition then action
  
  if car-in-front-is-braking then initiate-braking

- Rule conditions are compared with the current situation to determine which rule is applicable; once found, a rule is applied, meaning that some action is triggered.

- Environments need be fully observable
Simple reflex agents

A simple reflex agent works by finding a rule whose condition matches the current situation (as defined by the percept) and then doing the action associated with that rule.

```
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
```

Schematic diagram of a simple reflex agent.
Agents that keep track of the world (agents with internal states)

- In this case, rules are more complex; rule conditions take into account internal states of agents. By keeping some internal states, agents enhance their ability to react to environment. (To distinguish between states is fundamental in nature; recall states in logic design) Internal states capture info about how the world evolves and agent acts.

- Also called model-based agents

- Environments can be partially observable
Agents that keep track of the world (agents with internal states)

A reflex agent with internal state.

```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
```

A reflex agent with internal state. It works by finding a rule whose condition matches the current situation (as defined by the percept and the stored internal state) and then doing the action associated with that rule.
Goal-based agents

- Another enhancement is to give the agent a goal to look for. The agent actions constitute a sequence that leads to the goal.
  - Application examples: Searching, Planning.

- Goal information describes desirable situations.

- The actions now are not provided by if-then rules, but they are selected such that will bring the system closer to the goal.

- States are still necessary.

- Note that the way actions are selected to get closer to the goal, may still use if-then rules (but not exclusively).
Goal-based agents

A complete utility-based agent.
Utility-based agents

The idea is to provide a metric to assist the agent in moving toward the goal faster. A utility function maps the states into a performance number.
Learning Agents

A general model of learning agents.