How to Design an Intelligent Agent
About Intelligent Agents

- Agent comprises with
  - Environment
  - Agent
  - Percepts
  - Action
  - Sensors
Environment

• Determine to a large degree the interaction between the “outside world” and the agent
  – the “outside world” is not necessarily the “real world” as we perceive it
  – it may be a real or virtual environment the agent lives in

• in many cases, environments are implemented within computers
  – they may or may not have a close correspondence to the “real world
Environment types

- **Fully observable** (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.

- **Deterministic** (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is **strategic**)

- **Episodic** (vs. sequential): An agent’s action is divided into atomic episodes. Decisions do not depend on previous decisions/actions.
Environment types

- **Static** (vs. **dynamic**): The environment is unchanged while an agent is deliberating. (The environment is **semidynamic** if the environment itself does not change with the passage of time but the agent's performance score does)

- **Discrete** (vs. **continuous**): A limited number of distinct, clearly defined percepts and actions.

  How do we **represent** or **abstract** or **model** the world?

- **Single agent** (vs. **multi-agent**): An agent operating by itself in an environment. Does the other agent interfere with my performance measure?
Environment Properties

- Fully observable vs. partially observable
  - sensors capture all relevant information from the environment
- deterministic vs. stochastic (non-deterministic)
  - changes in the environment are predictable
- episodic vs. sequential (non-episodic)
  - independent perceiving-acting episodes
- static vs. dynamic
  - no changes while the agent is “thinking”
- discrete vs. continuous
  - limited number of distinct percepts/actions
- single vs. multiple agents
  - interaction and collaboration among agents
    - competitive, cooperative
Environment Programs

- Environment simulators for experiments with agents
  - gives a percept to an agent
  - receives an action
  - updates the environment

- Often divided into environment classes for related tasks or types of agents

- The environment frequently provides mechanisms for measuring the performance of agents
Task Environment

• Before we design an intelligent agent, we must specify its “task environment”:

• PEAS:
  – Performance measure
  – Environment
  – Actuators
  – Sensors
We now consider agents that **maintain state**: 

![Diagram showing the interaction between an agent, environment, and state](image)
Example

Vacuum cleaner robot
Solution
• Example: Agent = taxi driver

  – **Performance measure**: Safe, fast, legal, comfortable trip, maximize profits

  – **Environment**: Roads, other traffic, pedestrians, customers

  – **Actuators**: Steering wheel, accelerator, brake, signal, horn

  – **Sensors**: Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard
PEAS

- Example: Agent = Medical diagnosis system

  **Performance measure:** Healthy patient, minimize costs, lawsuits

  **Environment:** Patient, hospital, staff

  **Actuators:** Screen display (questions, tests, diagnoses, treatments, referrals)

  **Sensors:** Keyboard (entry of symptoms, findings, patient's answers)
PEAS

- **Example**: Agent = Part-picking robot

- **Performance measure**: Percentage of parts in correct bins

- **Environment**: Conveyor belt with parts, bins

- **Actuators**: Jointed arm and hand

- **Sensors**: Camera, joint angle sensors
<table>
<thead>
<tr>
<th>task env.</th>
<th>observable</th>
<th>determ./stochastic</th>
<th>episodic/sequential</th>
<th>static/dynamic</th>
<th>discrete/continuous</th>
<th>agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>crossword puzzle</td>
<td>fully</td>
<td>determ.</td>
<td>sequential</td>
<td>static</td>
<td>discrete</td>
<td>single</td>
</tr>
<tr>
<td>chess with 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>partial</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>partial</td>
<td>stochastic</td>
<td>sequential</td>
<td>dynamic</td>
<td>continuous</td>
<td>multi</td>
</tr>
<tr>
<td>medical diagnosis</td>
<td>partial</td>
<td>stochastic</td>
<td>sequential</td>
<td>dynamic</td>
<td>continuous</td>
<td>single</td>
</tr>
<tr>
<td>image analysis</td>
<td>fully</td>
<td>determ.</td>
<td>episodic</td>
<td>semi</td>
<td>continuous</td>
<td>single</td>
</tr>
<tr>
<td>partpicking robot</td>
<td>partial</td>
<td>stochastic</td>
<td>episodic</td>
<td>dynamic</td>
<td>continuous</td>
<td>single</td>
</tr>
<tr>
<td>refinery controller</td>
<td>partial</td>
<td>stochastic</td>
<td>sequential</td>
<td>dynamic</td>
<td>continuous</td>
<td>single</td>
</tr>
<tr>
<td>interact. Eng. tutor</td>
<td>partial</td>
<td>stochastic</td>
<td>sequential</td>
<td>dynamic</td>
<td>discrete</td>
<td>multi</td>
</tr>
</tbody>
</table>
Abstract Architecture for Agents

- Assume the environment may be in any of a finite set $E$ of discrete, instantaneous states:

  $$E = \{e, e', \ldots\}.$$ 

- Agents are assumed to have a repertoire of possible actions available to them, which transform the state of the environment:

  $$Ac = \{\alpha, \alpha', \ldots\}$$

- A run, $r$, of an agent in an environment is a sequence of interleaved environment states and actions:

  $$r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_{u-1}} e_u$$
Abstract Architecture for Agents

Let:

- $R$ be the set of all such possible finite sequences (over $E$ and $Ac$)
- $R^{Ac}$ be the subset of these that end with an action
- $R^E$ be the subset of these that end with an environment state
State Transformer Functions

• A state transformer function represents behavior of the environment:

\[ \tau : \mathcal{R}^{Ac} \rightarrow \wp(E) \]

• Note that environments are…
  – history dependent
  – non-deterministic

• If \( \tau(r) = \emptyset \), then there are no possible successor states to \( r \). In this case, we say that the system has ended its run

• Formally, we say an environment \( Env \) is a triple \( Env = \langle E, e_0, \tau \rangle \) where: \( E \) is a set of environment states, \( e_0 \in E \) is the initial state, and \( \tau \) is a state transformer function
Agents

- Agent is a function which maps runs to actions:

\[ \text{Ag} : \mathcal{R}_E \rightarrow \text{Ac} \]

An agent makes a decision about what action to perform based on the history of the system that it has witnessed to date. Let \( \text{AG} \) be the set of all agents.
Systems

• A *system* is a pair containing an agent and an environment

• Any system will have associated with it a set of possible runs; we denote the set of runs of agent $Ag$ in environment $Env$ by $\mathcal{R}(Ag, Env)$

• (We assume $\mathcal{R}(Ag, Env)$ contains only *terminated* runs)
Formally, a sequence

\[(e_0, \alpha_0, e_1, \alpha_1, e_2, \ldots)\]

represents a run of an agent \(Ag\) in environment \(Env = \langle E, e_0, \tau \rangle\) if:

1. \(e_0\) is the initial state of \(Env\)
2. \(\alpha_0 = Ag(e_0)\); and
3. For \(u > 0\),

\[e_u \in \tau((e_0, \alpha_0, \ldots, \alpha_{u-1}))\] where

\[\alpha_u = Ag((e_0, \alpha_0, \ldots, e_u))\]
Purely Reactive Agents

- Some agents decide what to do without reference to their history — they base their decision making entirely on the present, with no reference at all to the past.
- We call such agents *purely reactive*:

\[
\text{action} : E \rightarrow Ac
\]

- A thermostat is a purely reactive agent.

\[
\text{action}(e) = \begin{cases} 
\text{off} & \text{if } e = \text{temperature OK} \\
\text{on} & \text{otherwise.}
\end{cases}
\]
Software agent - Design

• Why Agent than the Objects
• Get Idea about Environment
• Input
• Output
• Processes
• Communication/ Action
• How to provide Intelligent capabilities
## Example

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Percepts</th>
<th>Actions</th>
<th>Goals</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medical diagnostic system</td>
<td>Symptoms, test results, patient’s answers</td>
<td>Questions, test requests, treatments, referrals</td>
<td>Healthy patients, minimise costs</td>
<td>Patient, hospital, staff</td>
</tr>
<tr>
<td>Satellite image analysis system</td>
<td>Pixels of varying intensity and colour</td>
<td>Display a categorisation of the scene</td>
<td>Correct image categorisation</td>
<td>Images from orbiting satellite</td>
</tr>
<tr>
<td>Part-picking robot</td>
<td>Pixels of varying intensity and colour</td>
<td>Pick up parts and sort them into bins</td>
<td>Place parts into correct bins</td>
<td>Conveyor belt with parts, bins</td>
</tr>
<tr>
<td>Refinery controller</td>
<td>Temperature, pressure and chemical readings</td>
<td>Open and close valves, adjust temperature</td>
<td>Maximise purity, yield, safety</td>
<td>Refinery, staff</td>
</tr>
<tr>
<td>Interactive English tutor</td>
<td>Typed words</td>
<td>Display exercises, suggestions, corrections</td>
<td>Maximise student’s exam results</td>
<td>Set of students, exam papers</td>
</tr>
</tbody>
</table>