EECS 391: Introduction to AI

Soumya Ray
Website: http://engr.case.edu/ray_soumya/eecs391_sp15/
Email: sray@case.edu
Office: Olin 516
Office hours: F 11:30-2pm or by appointment
Announcements

• SEPIA documentation online
• HW1 out later today
• Office hours: F 11:30-2pm
  – 11:30-12:30 I will be available
  – 12:30-2pm I may occasionally be unavailable
  (send email to check)
Today

• General Architecture of Intelligent Agents (Chapter 2)
• Uninformed Search (Chapter 3)
Basic Agent Architecture

• Agent = something that interacts with the world

“Ai Function” $A$: Percept Sequences $\rightarrow$ Actions
Examples

• Chess-playing agent
  – Sensors?
  – Actuators?

• Autonomous vehicle agent
  – Sensors?
  – Actuators?
Performance Measures

• Agents are usually “goal-based”, i.e. they are designed to achieve certain things in the world
  – Chess-playing agent?
  – Autonomous vehicle agent?
• These are often encoded using a “performance measure”
  – A function that maps a (percept, action) sequence to a real number
  – Generally externally imposed
  – Can think of this as an internal “satisfaction” or “reward” signal
Rational Agents

• A rational agent is one whose agent function always acts to maximize its performance measure, given its percept sequence until the current moment.
Example

• Suppose the autonomous vehicle agent is given the measure: +1 point every second without a collision
  – What might a rational agent do?
“PEAS” description

• To design the (rational) agent function we need four things:
  – The **Performance** measure
  – A description of the **Environment**
  – A description of what **Actions** the agent has
  – A description of what **Sensors** the agent has
Examples

• Chess-playing agent PEAS?

• Autonomous vehicle agent PEAS?
# Types of Environments

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully observable (vs. Partially Observable)</td>
<td>Agent’s sensors present complete, accurate picture of the world (as far as determining action sequence is concerned)</td>
</tr>
<tr>
<td>Deterministic (vs. Stochastic)</td>
<td>The next state of the world is completely determined by current state and agent’s action</td>
</tr>
<tr>
<td>Non-sequential (Episodic) (vs. Sequential)</td>
<td>Agent’s current action does not affect future actions</td>
</tr>
<tr>
<td>Static (vs. Dynamic)</td>
<td>The world does not change until the agent takes an action</td>
</tr>
<tr>
<td>Discrete (vs. Continuous)</td>
<td>States, percepts and actions are discrete</td>
</tr>
<tr>
<td>Single Agent (vs. Multiagent)</td>
<td>The world has only one agent in it</td>
</tr>
</tbody>
</table>
Example

• Environment of chess-playing agent?

• Environment of autonomous vehicle agent?
Types of Agent Functions

• Simple Reflex

• Model-based Reflex

• Goal-based

• Utility-based
Simple Reflex Agents

• Agent function maps current state directly to action

Would this work for the chess agent? For the vehicle agent?
Model-based Reflex Agent

• Maps current percept and “world model” to action

Condition-action rules:
If $State = x$ perform action $a$

World model

State Estimator

Sensors

Percepts

World/Environment

Actuators

Action
Goal-based Agent

• Maps current percept and knowledge of current goal to action

World model

State Estimator

Goal-based action choice
If State == x and current-goal == g
perform action a

Sensors

Percepts

World/Environment

Actuators

Action
Utility-based Agent

- Instead of binary goal, an agent could have a fine-grained notion of how useful certain states are.

Utility-based action choice:
If State $= x$ and $\text{NextState}(x,a) = y$
and $\text{Utility}(y)$ is high
perform action $a$
Learning Agents

• Who writes all these rules?
  – Or designs the agent function?
  – Could be quite complex!

• Could we have the agent learn the agent function on its own?
  – Would also help in unknown environments...
General Architecture

• Before, we had fixed rules (or functions)
  – We’ll call this the “performance element”

• Now we need to add something that generates those functions, based on its (partial) knowledge of the environment and any feedback it receives
  – We’ll call this the “learning element”
General Architecture

Performance

“Critic”

Learning Element

Changes

Performance Element

Problem Generator

Actuators

Exploratory Goals

World/Environment

Percepts

Action
Summary

• We learned about:
  – Agent architecture
  – PEAS descriptions
  – Types of environments
  – Types of Agents
Solving Problems Using Goal-Directed Search (Chapter 3)

• Idea
• How to set up the problem
• Basic algorithms
• Characteristics of algorithms
Overview

- We saw that an intelligent agent needs to be flexible.
- So we can’t give it solutions to specific problems, we need to give it *problem solving strategies*.
- The most basic general purpose problem solving strategy is called “Goal-Directed Search”.

1/18/2015 Soumya Ray, Case Western Reserve U.
Goal-Directed Search

• A fundamental technique in AI

• A strategy when the agent has limited/no idea about the detailed structure of a problem to allow more complex reasoning

• Very easy to implement

• Will show up often when we study the more complex algorithms later
When to use Goal-Directed Search

• The agent *fully perceives* the current state of the world and wants to achieve a certain goal state
  – It can distinguish different states
  – In particular, goal situations from non-goal situations
  – It can tell how the state will change if it takes an action (e.g. “state 5 will become state 43 if I go left”)

• It wants the *least cost path* to get to the goal
“Offline” Problem Solving

• The entire search operation is part of the “agent function”---it is internal to the agent

• After the search is complete and the solution found, the agent can apply them to the world to execute the solution
  – World needs to be static
Environment Type

• We’ll assume the environment is:
  – Fully observable (to track the state)
  – Static (shouldn’t change while agent is searching)
  – Deterministic (agent needs to be able to precisely predict states resulting after each action)

• Some search algorithms also need discrete environments
Examples

• Route Finding
  – Suppose an agent wants to get from location A to location B
  – Same techniques used by mapping software e.g. Google Maps and GPS systems

• Solving puzzles
  – 8-puzzle
  – Sudoku