Agents: Intro

• Agent is entity that perceives and acts
  – Perception occurs via sensors
    * Percept is one unit of sensory input
    * Percept sequence is complete history of agent’s percepts
    * In general, agent’s actions are based only on what has been perceived
      · If agent receives no percepts, will not be able to adapt to environment
  – Action accomplished via effectors/actuators

• Agent is a function that maps percept sequences to actions
  – Mapping can be thought of in two general ways:
    1. A lookup table indexed by percept sequences (external representation)
    2. Algorithmically, generating an action given a percept sequence as input
       (internal representation)
      * Implemented in terms of an agent program
Agents: Rationality

• Concerned with the construction of intelligent (rational) agents
  – Rational agent is agent that ”does the right thing”
    * Need objective measures of what is ”right” and how successful the agent is in achieving it

• Performance measure: a criterion for success
  – Used to evaluate environmental states that result from agent’s actions
  – Must differentiate between an agent’s internal state and the state of the environment
  – Criteria for performance measures:
    * Should be objective
    * Should be defined by outside agency
    * Must be careful \textit{what} is being measured
    * Must be careful \textit{when} it is measured
    * Success should be an average of performance over an extended period

• Rationality depends on
  1. Percept sequence
  2. Knowledge about its environment
  3. Set of available actions
  4. Agent’s percept sequence

• Rational agent: For each possible percept sequence, a rational agent should perform whatever action is expected to maximize its performance measure for each possible percept sequence on the basis of the evidence provided by its percept sequence and whatever built-in knowledge it has available.

• Rationality is not the same as omniscience
  – \textit{Omniscience} means maximizing \textit{actual} results of actions
  – \textit{Rationality} means maximizing \textit{expected} results of actions

• Information gathering: Retrieving info from the environment (via actions) to help maximize performance
Agents: Rationality (2)

• Want *autonomous* agent
  
  – Autonomous agent can act on its own under its own control
  – Implies that the agent must be able to learn through its percepts
  – Autonomy does not preclude *a priori* knowledge, but it does preclude *a priori* knowledge *exclusively*
    * If agent operates exclusively on *a priori* knowledge, it will not be able to adapt to novel situations
Agents: Environments

• *Task environment* describes problem that agent solves
  – Consists of
    1. Performance measure
    2. Environment
    3. Actuators
    4. Sensors
  – Called the PEAS description
  – Interacts with the environment
  – Internal state constructed exclusively from percepts

• Environments can be described in terms of the following characteristics
  1. Fully observable v partially observable
    – Fully observable environment is one in which agent can perceive its complete state (those aspects relevant to agent’s actions)
    – No need to maintain an internal state of such an environment in order to keep track of it
  2. Single agent v multi agent
    – The difference lies in whether other entities in the environment have performance measures that depend on the agent’s behavior
    * If they do, the environment is multi agent
    * *Competitive* environment is one in which one agent maximizes its performance measure to the detriment of another agent’s performance measure
    * *Cooperative* environment is one in which maximization of one agent’s performance measure helps maximize another agent’s performance measure
3. Deterministic v stochastic
   - In deterministic environment, next state is determined solely by current state and action of the agent
   - No need to worry about uncertainty
   - *Uncertain* environment is one that is not fully observable or not deterministic
   - *Nondeterministic* environment is one in which actions characterized by possible outcomes but have no probabilities associated with the outcomes
     * Such environments usually require agent to search through all possible outcomes of actions (exhaustive search)

4. Episodic v sequential
   - *Episode* is a single action independent of others, based on a single percept
   - In episodic environment results of one episode do not affect other episodes
   - Agent does not need to plan ahead (wrt episodes)
   - Sequential environment is one in which an action (may) influence future actions

5. Static v dynamic
   - Static environment is changed only by the agent’s actions
     * It does not evolve independently of the agent
     * The agent can take its time making decisions and not need to worry about the world evolving as it deliberates
   - *Semidynamic* environment is one that is static but where the performance measure is time-dependent

6. Discrete v continuous
   - Difference based on
     (a) Environment state
     (b) How time is handled
     (c) Percepts
     (d) Actions
   - Discrete environment is one in which only a distinct number of each of the above exist
7. Known v unknown

- This applies the the agent’s internal state model
- An unknown environment is one in which the agent does not know the rules of play
  * Unknown does not mean partially observable

- The most difficult environment is partially observable, multiagent, stochastic, sequential, dynamic, continuous, and unknown
  - This describes most real-world problems

- Environment simulator program:

```plaintext
procedure Run-Environment (state, Update-Function, agents, termination)
{
  input: state //description of initial world state
  //this distinct from agent’s state and inaccessible to agent
  Update-Function //modifies environment
  agents
  termination //predicate for termination

  repeat
    for each agent do
      Percept(agent) <- Get-Percept(agent, state)
    for each agent do
      Action(agent) <- Program(agent)(Percept(agent))
      state <- Update-Function(actions, agents, state)
    until termination(state)
}
```

- Most agents designed to work in an environment class - set of environments
  - Must evaluate agent’s performance in a variety of environments
  - If it only performs in one, it may take advantage of the environment’s weaknesses and so performance measure is biased
Agents: Agent Programs

- Goal of recent AI is to construct *agent programs* - the programs that map percept sequences to actions

  - To design such a program, need to establish the platform on which it will run
    * An agent consists of a program and an architecture
    * To design the platform, need to know things like
      1. Types of percepts needed (or available)
      2. Actions needed (or available)
      3. Goals and performance measures
      4. Environment
    * The above is referred to as a *PAGE* description
    * These implemented in terms of
      1. Sensors (physical or soft)
      2. Effectors (physical or soft)
      3. Computing architecture
      4. The program itself

- Basic agent structure (function):

  ```
  function BasicAgent (percept) returns action 
  {
    static memory

    memory <- UpdateMemory(memory, percept)
    action <- ChooseBestAction(memory)
    memory <- UpdateMemory(memory, action)
    return action
  }
  ```
Agents: Types - Table-driven Agents

- Simplest agent program
- Agent program:

```plaintext
function Table-Driven-Agent (percept) returns action
{
    static perceptSequence  //a sequence, initially empty
    table                   //indexed by percept sequences

    append(percept, perceptSequence)
    action <- Lookup(perceptSequence, table)
    return action
}
```

- Issues:
  1. Size of table
  2. Construction of table
     - Must be exhaustive
     - Time involved
  3. Not autonomous
     - Cannot adapt to novel situations
- Above arguments are for use of a lookup program exclusively
  - There is nothing to preclude table lookup as part or in conjunction with another approach
Agents: Types - Simple Reflex Agents

- Structure:

  ![Diagram of a simple reflex agent](image)

  - Agent program:

    ```
    function Simple-Reflex-Agent (percept) returns action
    {
        static rules

        state <- Interpret-Input(percept)
        rule <- Rule-Match(state, rules)
        action <- rule.Action
        return action
    }
    ```

  - Agent does not maintain an evolving representation of the world
    - State is a representation based on current percept - a snapshot
    - There is no memory
    - Each action based solely on current percept

  - Uses a specialized lookup table (`rules`) for situations that require no significant deliberation
Agents: Types - Simple Reflex Agents (2)

- Individual rules take the form of *if-then* statements
  - Referred to as
    * *Condition-action* rules
    * *Situation-action* rules
    * *If-then* rules
    * *Productions*

- Issues:
  1. No internal representation of the world
    - Cannot make predictions about how the world may change
    - Cannot plan sequences of actions
    - Cannot base decisions on time-varying data
  2. Prone to infinite loops in partially-observable environments
    - Can be avoided by incorporating randomness when choose rules
Agents: Types - Model-based Reflex Agents

• Structure:

```
function Model-based-Reflex-Agent (percept) returns action {
    static state  //description of current world state
    model  //state of world as result of action after applied to current state
    rules
    action  //most recent action

    state <- Update-State(state, action, percept, model)
    rule <- Rule-Match(state, rules)
    action <- rule.Action
}
```

• Uses a specialized lookup table (rules) for situations that require no significant deliberation
Agents: Types - Model-based Reflex Agents (2)

• Assumes evolving world
  – Changes my result from
    * Agent’s actions
    * External forces

• Cannot assume complete knowledge about the world
  – Sensors cannot monitor all aspects of the world

• Agent will utilize knowledge about how the world works
  1. Knowledge re how the agent’s actions affect the world
  2. Knowledge about how the world evolves

• Issues:
  1. No mechanism for making decisions
     – Cannot decide among several alternative actions
Agents: Types - Goal-based Agents

- Structure:

- Incorporation of goals allows agent to use them to decide among conflicting actions
  - In reflexive agents, no reasoning is involved
  - If a rule’s conditions are met, the action is carried out

- Must be able to predict results of actions
  - Use these results to determine which action takes agent closer to goal
  - Incorporates search and planning

- Issues:
  1. Slower than reflexive agents
     - Must reason about effects of actions
Agents: Types - Goal-based Agents (2)

- More flexible than reflexive agents
  - When a novel situations arises, a goal-based agent can reason about the effects
    * In reflexive case, would need to modify rule base to account for the novel situation
  - Can work with a variety of goals
    * In reflexive case, would need to modify rule base to accomodate new goals
Agents: Types - Utility-based Agents

- Structure:

- Goal-based agents lack a "happiness" factor
  - There are usually many paths (sequences of actions) that can lead to a goal
  - There should be some measure of the desireability of states that the agent passes through on its path to a goal
  - Desireability might be measured in terms of
    * Efficiency
    * Cost (energy, damage, etc.)
    * Directness
Agents: Types - Utility-based Agents (2)

- *Utility* measures the desireability of a state
- *Utility function* maps a state to a number
  - Can be used in several ways:
    1. To select from among conflicting goals
      * Allows intelligent decision making with respect to tradeoffs
    2. In uncertain environments, allows intelligent choices based on the likelihood of success
- Utility-based agent may need to compare the utilities that arise from different courses of action
  - Goal-based agent will simply select an action if it achieves a goal
- Since world is generally an uncertain place, agent will choose action that maximizes *expected utility*
  - Based on the probabilities and utilities associated with outcomes of actions
Agents: Types - Learning Agents

• To create a rational agent
  1. Could program all aspects of intelligence re a given environment by hand
     – Not practical
  2. Could incorporate a learning component and basic knowledge, and let the agent learn
     – Reduces amount of programming involved
     – Makes a more flexible agent

• Structure:
Learning agent has four components

1. Learning element
   – Responsible for making improvements
   – Can alter any of the data structures (state info) of the agent

2. Performance element
   – Embodies the agent program

3. Critic
   – Evaluates agent’s performance
   – Passes recommendations to the learning element

4. Problem generator
   – Suggests actions that will force the agent to explore
   – Purpose is to force agent to continue learning in hopes of improving performance
     * Otherwise the agent would continue performing actions that had known success

Performance standard

– Should be fixed and external to the agent
– Want the agent’s performance to be based on an absolute standard and not based on the agent’s internal perception of its performance
– It may (partially) interpret some percepts as rewards or punishments to be used as feedback
Agents: Agent Program State Structure

- Internal state can be represented in three general ways

1. Atomic
   - State represented as an atomic value
   - Has no internal structure

2. Factored
   - State consists of a set of variables that represent attributes of the environment
   - Each has a value
   - Allows representation of uncertainty (associated with each attribute)
   - Attributes may be common to several states

3. Structured
   - Includes relationships among the attributes