Structure of Intelligent Agents

An agent:

- Perceives its environment,
- Through its sensors,
- Then achieves its goals
- By acting on its environment via actuators.
Examples of Agents 1

- **Agent**: mail sorting robot
- **Environment**: conveyor belt of letters
- **Goals**: route letter into correct bin
- **Percepts**: array of pixel intensities
- **Actions**: route letter into bin

Side info: https://en.wikipedia.org/wiki/Mail_sorter
Examples of Agents 2

- **Agent**: intelligent house
- **Environment**: 
  - occupants enter and leave house,
  - occupants enter and leave rooms;
  - daily variation in outside light and temperature
- **Goals**: occupants warm, room lights are on when room is occupied, house energy efficient
- **Percepts**: signals from temperature sensor, movement sensor, clock, sound sensor
- **Actions**: room heaters on/off, lights on/off

Examples of Agents 3

- **Agent**: autonomous car.
- **Environment**: streets, other vehicles, pedestrians, traffic signals/lights/signs.
- **Goals**: safe, fast, legal trip.
- **Percepts**: camera, GPS signals, speedometer, sonar.
- **Actions**: steer, accelerate, brake.

Side info: https://en.wikipedia.org/wiki/Autonomous_car
Simple Reflex Agents

- Action depends only on immediate percepts.
- Implement by condition-action rules.
- Example:
  - **Agent**: Mail sorting robot
  - **Environment**: Conveyor belt of letters
  - **Rule**: e.g. $city=Edinburgh \rightarrow put Scotland bag$

**Simple Reflex Agents**

**function** SIMPLE-REFLEX-AGENT(*percept*)
returns *action*

**persistent**: *rules* (set of condition-action rules)

  *state* ← INTERPRET-INPUT(*percept*)
  *rule* ← RULE-MATCH(*state*, *rules*)
  *action* ← *rule*.ACTION

**return** *action*
Model-Based Reflex Agents

- Action may depend on history or unperceived aspects of the world.
- Need to maintain internal world model.
- Example:
  - **Agent**: robot vacuum cleaner
  - **Environment**: dirty room, furniture.
  - **Model**: map of room, which areas already cleaned.
  - Sensor/model trade-off.
function REFLEX-AGENT-WITH-STATE(percept)
  returns action
  persistent: state, description of current world state
              model, description of how the next state depends on
              current state and action
              rules, a set of condition-action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION

  return action
Goal-Based Agents

- Agents so far have fixed, implicit goals.
- We want agents with variable goals.
- Forming plans to achieve goals is later topic.
- **Example:**
  - **Agent:** household service robot
  - **Environment:** house & people.
  - **Goals:** clean clothes, tidy room, table laid, etc.
Utility-Based Agents

- Agents so far have had a single goal.
- Agents may have to juggle conflicting goals.
- Need to optimise utility over a range of goals.
- **Utility**: measure of *goodness* (a real number).
- Combine with probability of success to get *expected utility*.

**Example:**
- **Agent**: automatic car.
- **Environment**: roads, vehicles, signs, etc.
- **Goals**: stay safe, reach destination, be quick, obey law, save fuel, etc.
Utility-Based Agents

We will not be covering utility-based agents, but this topic is discussed in Russell & Norvig, Chapters 16 and 17.
Learning Agents

How do agents improve their performance in the light of experience?

- Generate problems which will test performance.
- Perform activities according to rules, goals, model, utilities, etc.
- Monitor performance and identify non-optimal activity.
- Identify and implement improvements.

We will not be covering learning agents, but this topic is dealt with in several honours-level courses (see also R&N, Ch. 18-21).
Consider a chess playing program.

What sort of agent would it need to be?
Simple-reflex agent: but some actions require some memory (e.g. castling in chess: http://en.wikipedia.org/wiki/Castling).

Model-based reflex agent: but needs to reason about future.

Goal-based agent: but only has one goal.

Utility-based agent: might consider multiple goals with limited lookahead.

Learning agent: Learns from experience or self-play
Fully Observable vs. Partially Observable:
- Full: agent’s sensors describe environment state fully.
- Partial: some parts of environment not visible, noisy sensors.

Deterministic vs. Stochastic:
- Deterministic: next state fully determined by current state and agent’s actions.
- Stochastic: random changes (can’t be predicted exactly).

An environment may appear stochastic if it is only partially observable.
Types of Environment 2

- **Episodic vs. Sequential:**
  - Episodic: next action does not depend on previous actions.
  - Mail-sorting robot vs. crossword puzzle.

- **Static vs. Dynamic:**
  - Static: environment unchanged while agent deliberates.
  - Crossword puzzle vs. chess.
  - Industrial robot vs. robot car
Discrete vs. Continuous:
- Discrete: percepts, actions and episodes are discrete.
- Chess vs. robot car.

Single Agent vs. Multi-Agent:
- How many objects must be modelled as agents.
- Crossword vs. poker.

Element of choice over which objects are considered agents.
An agent might have any combination of these properties:

- from “benign” (i.e., fully observable, deterministic, episodic, static, discrete and single agent)
- to “chaotic” (i.e., partially observable, stochastic, sequential, dynamic, continuous and multi-agent).

What are the properties of the environment that would be experienced by

- a mail-sorting robot?
- an intelligent house?
- a car-driving robot?
Summary

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents
- Learning agents
- Properties of environments