Artificial Intelligence

Intelligent Agents

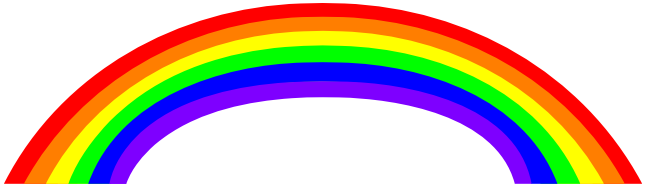
Lecture 2

(August 18, 1999)

Tralvex (Rex) Yeap MAAAI MSCS

University of Leeds

Content: Intelligent Agents



- Quick Review on Lecture 1
- Intelligent Agents
  - Three laws of robotics
  - What is an Agent
  - How Agents should Act
  - Rational behaviour depends on knowledge
  - Structure of an Intelligent Agent
  - Examples of Agents and their PAGE description
  - Five Major Agent Types
  - Shopping Example Activities
  - Agent Environments
  - An Agent Portfolio
- Class Activity 1: To write the PAGE description for Robocup
- Class Activity 2: To write the characteristics of the environment in Robocup domain.
- Class Activity 3: A paper on Intelligent Agent - Reading.
- What's in Store for Lecture 2b
- What's in Store for Lecture 3

Quick Review on Lecture 1

- N-ways Introduction
- Course Outline
- Introduction to AI
  - What is Intelligence?
  - An Intelligent Entity
  - The Age of Intelligent Machines
  - Definitions of AI
  - Behaviourist's View on Intelligent Machines
  - Turing's Test - Part 1 & 2
  - History of AI
  - Examples of AI systems
- AI Case Study
  - Radiosity for Virtual Reality Systems
  - Robot World Cup - Robocup
  - Scrablet
- Class Activity: AI & Us

Introduction

The Three Laws of Robotics

1. A robot **may not injure** a human being, or, through inaction allow a human being to come to harm.

2. A robot must **obey the orders** given it by human beings except where such orders would conflict with the First Law.

3. A robot must **protect its own existence** as long as such protection does not conflict with the First or Second Law.

Handbook of Robotics. 56th Edition, 2058 A.D.
From Isaac Asimov.

What is an Agent

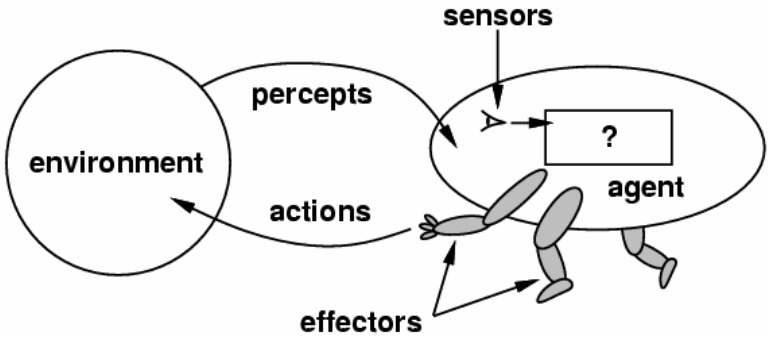
- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

- A Human
  Sensors: eyes, ears, nose, tongue, skin, ...
  Effectors: hands, legs, mouth, ...

- A Robot
  Sensors: video cameras, infrared range finder, ...
  Effectors: motors, wheels, ...

- A Software Agent:
  Sensors: encoded bit strings, ...
  Effectors:encoded bit strings, ...
  ➢ Concept of virtual sensors and effectors

- Goal: Design agents that do a good job of acting in their environments.



How Agents should Act

- "For each possible **percept sequence**, an ideal rational agent **should do** whatever action is expected to maximize its **performance measure**, on the basis of the **evidence** provided by the **percept sequence** and whatever **built-in knowledge** the agent has"

Russell, Norvig (1995), page 33.

How Agents should Act (cont)

- **Rational** = based on **reason**
- Given options, chooses the **best for "success"**
- **Objective performance** measures needed:
  - Succeed at task?
  - Resources consumed?
  - Time taken?

- **Automated driver**
  - reach destination?
  - safe, legal driving
  - travel minimum distance to destination?
- **Dirt-cleaning robot**
  - Amount of dirt in 8 hour shift
  - Amount of electricity consumed & noise generated
  - How clean the floor is

Rational behavior depends on knowledge

- Cannot expect omniscience (having infinite knowledge or God)
  Example: Driver chooses most direct route on map. Turns out route is blocked for road work. Wasted time, should have gone other way. Limited perception.

- Reasonable expectation of agent
  Only take into account what it can perceive; limited selection of actions; obliged to look.

- Behavior depends on the mapping **from percept sequences to actions**:

  ➢ **Ideal mapping**
  "...which action an agent ought to take in response to any given percept sequence..."

  ➢ **Table look-up**
  Could be infinitely long unless a bound is put on length of percept sequence.

  ➢ **Function**
  Does not require exhaustive enumeration. e.g., square-root.

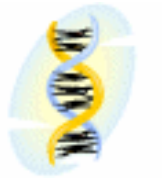
Rational behavior depends on knowledge (cont)

- **Autonomous behavior**:
  - what about agents that work completely from knowledge and don't pay attention to percepts?
  - **autonomy** requires an agent to **change behavior based on percepts**
  - ideally, the agent **learns from experience** and **adapts behavior** accordingly--change the mapping from percept sequence to actions
  - Sometimes the agent designers are the ones who learn from experience and adapt the system!

Structure of an Intelligent Agent

- **agent = architecture + program**
- program: a function that maps percepts to actions
- architecture:
  - where the program runs: HW+SW
  - makes percepts available to program
  - runs program
  - feeds actions from program to effectors
- For each agent we must analyze its (1) **Percepts** (2) **Actions** (3) **Goals** and (4) **Environment**, in short PAGE.

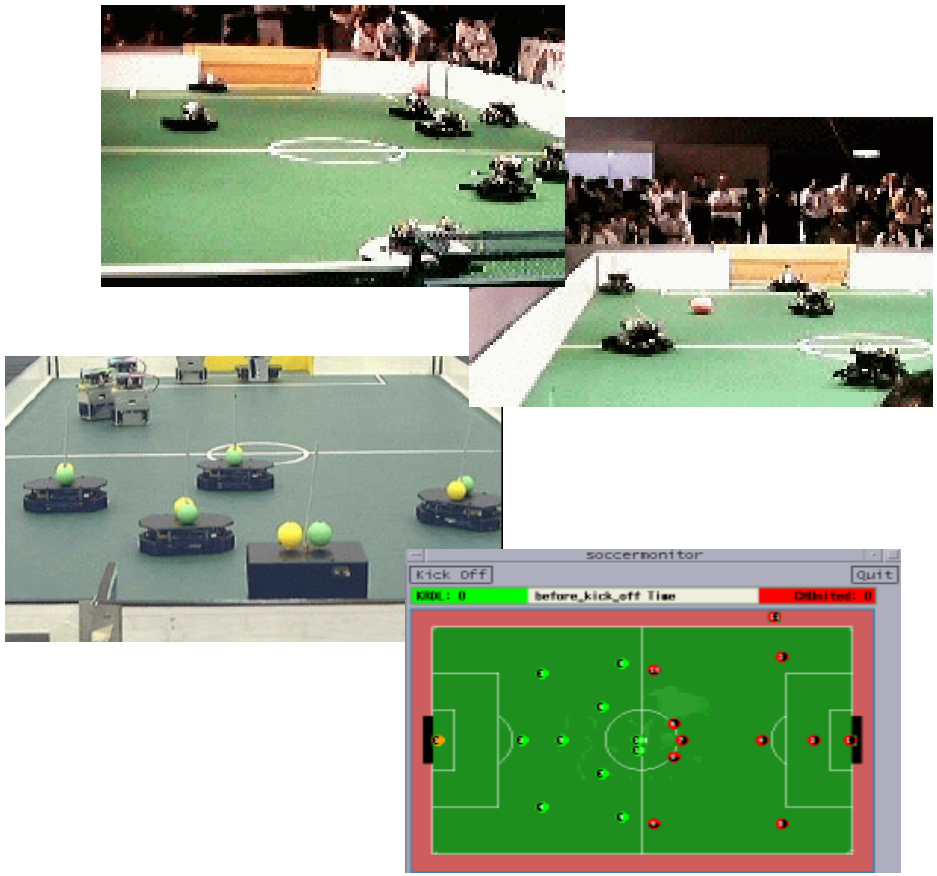
Examples of Agents and their PAGE Description



Agent	Percepts	Actions	Goals	Environment
Part-picking robot	Pixels of varying intensity	Pick up parts and sort into bins	Place parts in correct bins	Conveyer belt with parts
Refinery controller	Temperature, pressure readings	Open, close valves; adjust temperature	Maximize purity, yield, safety	Refinery
Satellite image analysis system	Pixels of varying intensity, color	Print a categorization of scene	Correct categorization	Images from orbiting satellite
Medical diagnosis system	Symptoms, findings, patient's answers	Questions, tests, treatments	Healthy patient, minimize costs	Patient, hospital
Automated Driver	Cars, lights (pixels) GPS,speedometer radio, keyboard	Steer, accelerate, brake	Safe,fast,legal, minimize mileage	Roads, traffic, pedestrians
Shopper Robot	Groceries, signs, carts, etc.	Navigate, gather, move	Get all items, minimize cost, mininmize distance	Supermarket

Class Activity 1: To write the PAGE description for Robocup

- **Percepts**
- **Actions**
- **Goals**
- **Environment**



Major Agent Types

- **Table-driven agents** use a percept sequence/action **table in memory** to find the next action. They are implemented by a (large) lookup table.

- **Simple reflex agents** are based on condition-action rules and implemented with an appropriate **production system**. They are stateless devices which **do not have memory** of past world states.

- **Reflex agents with state** have **internal memory** which is used to keep track of past states of the world.

Major Agent Types (cont)

- **Agents with goals** are agents which in addition to state information have a kind of **goal information** which describes **desirable situations**. Agents of this kind. take **future events** into consideration.

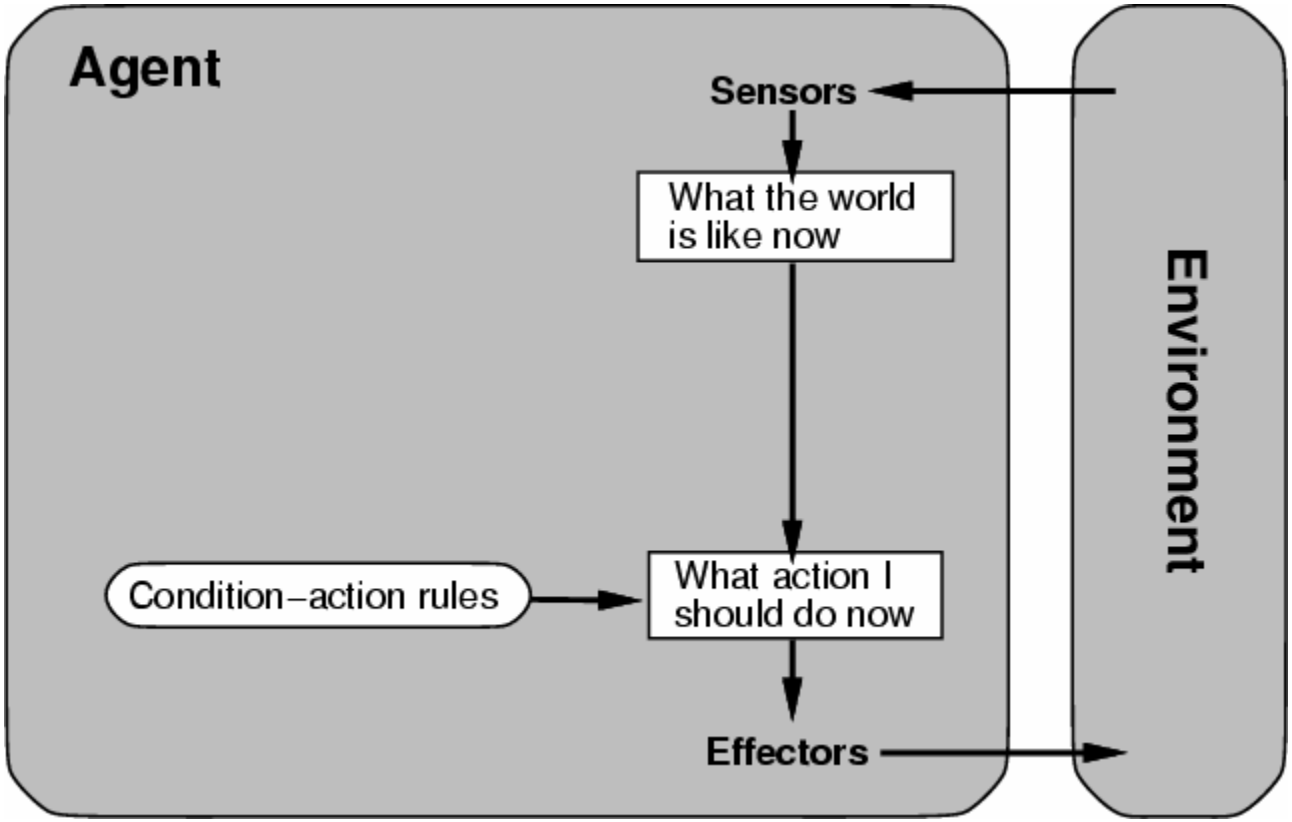
- **Utility-based agents** base their decision on classic utility-theory in order to act rationally.

Major Agent Types
(1) Table-driven Agents


```
function TABLE-DRIVEN-AGENT(percept) returns action
    static: percepts, a sequence, initially empty
            table, a table, indexed by percept sequences, initially fully specified

    append percept to the end of percepts
    action ← LOOKUP(percepts, table)
    return action
```

- Why this proposal is doomed to fail?
  - The table needed for something as simple as an agent that can only play chess would be about **35^{100} entries**.
  - It would take quite a **long time** for designer to **build** the table.
  - The agent has **no autonomy** at all, because the calculation of best actions is entirely built-in. So if the **environment changed** in some unexpected way, the **agent would be lost**.

Major Agent Types
(2) Simple Reflex Agents

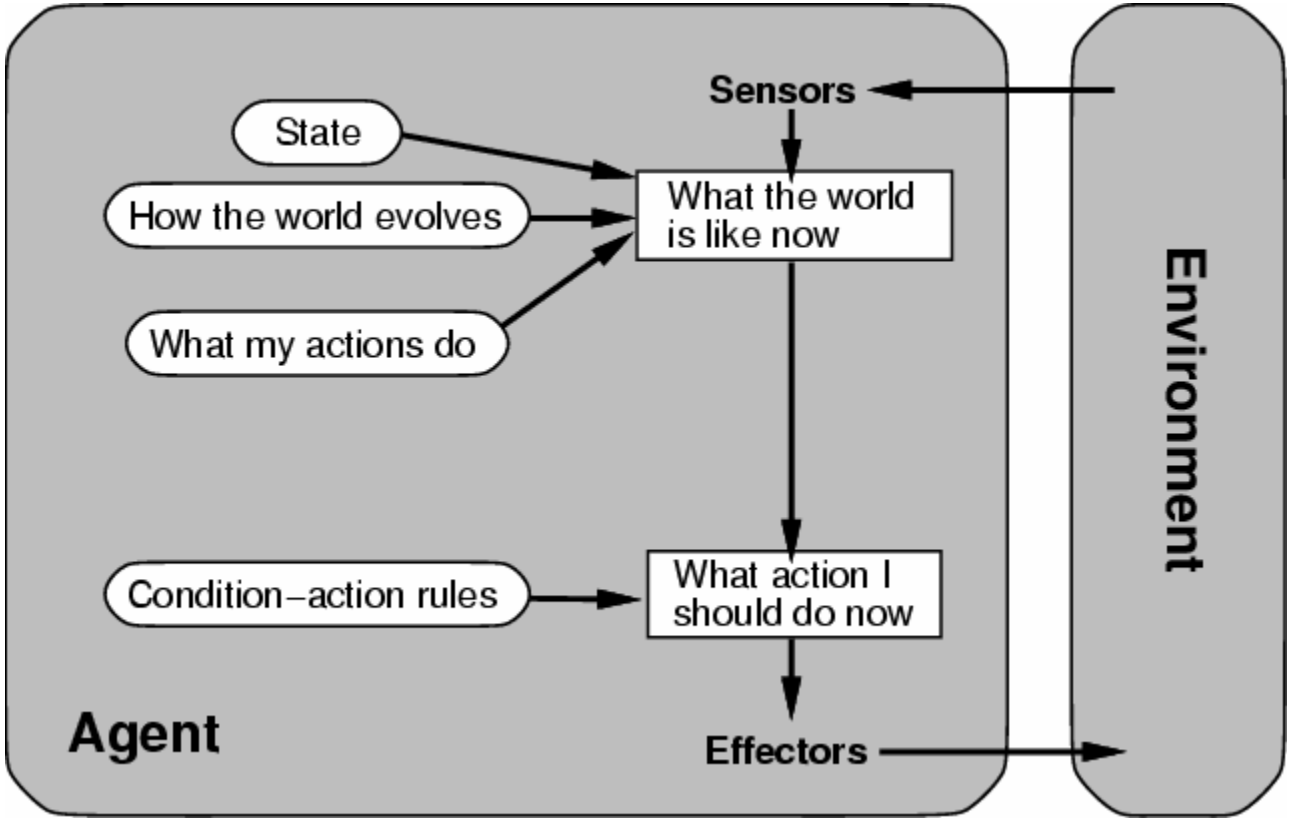


Major Agent Types
(2) Simple Reflex Agents (cont)

```
function SIMPLE-REFLEX-AGENT(percept) returns action
   static: rules, a set of condition-action rules

   state ← INTERPRET-INPUT(percept)
   rule ← RULE-MATCH(state, rules)
   action ← RULE-ACTION[rule]
   return action
```

- Uses **condition-action rules**, productions. In humans, condition-action rules are both learned responses and innate reflexes (e.g., blinking).
  - if light-is-green then accelerate
  - if light-is-red then brake
- Correct decisions must be made solely based on **current percept**.
- **Counter** examples:
  - Is driving purely a matter of reflex? What happens when making a lane change? Must remember what you saw in the rear-view mirror.
  - Grocery shopping: if milk-carton then put in basket. What happens when you get to the dairy section? Can't remember what is already in the basket.

Major Agent Types
(3) Reflex Agents with State

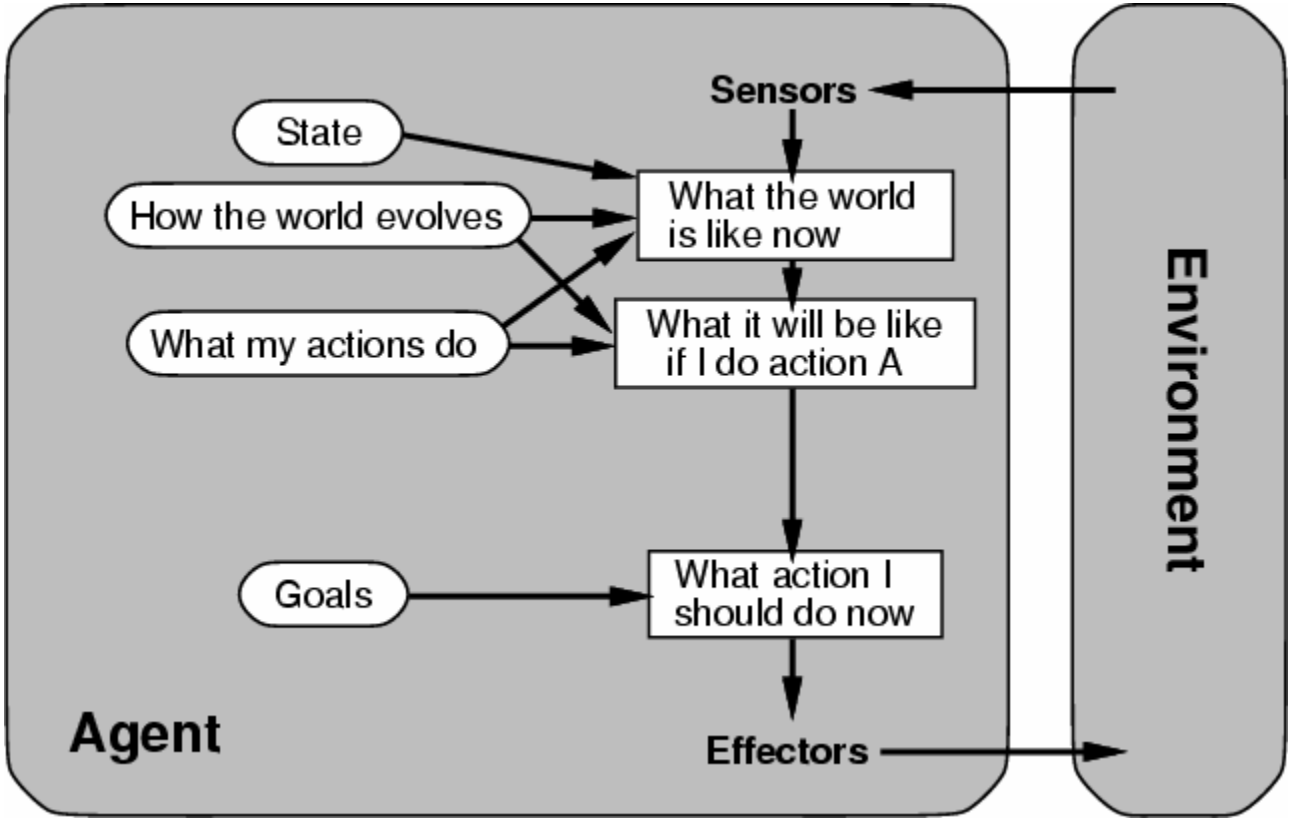


Major Agent Types
(3) Reflex Agents with State (cont)

```
function REFLEX-AGENT-WITH-STATE(percept) returns action
  static: state, a description of the current world state
          rules, a set of condition-action rules

  state ← UPDATE-STATE(state, percept)
  rule ← RULE-MATCH(state, rules)
  action ← RULE-ACTION[rule]
  state ← UPDATE-STATE(state, action)
  return action
```

- **Internal state** is used to distinguish between world states that generate same immediate percept but are different (because of past action) with respect to the action to be taken. e.g., Standing in the dairy section, see other milk cartons.
- Requires ability to **represent change in the world**: how the world evolves and how the actions affect the world.
- One possibility is to represent just the **latest state** if that is all that matters - concept of **ghost states** (eg. in Robocup domain)**.**

Major Agent Types
(4) Agents with Goals



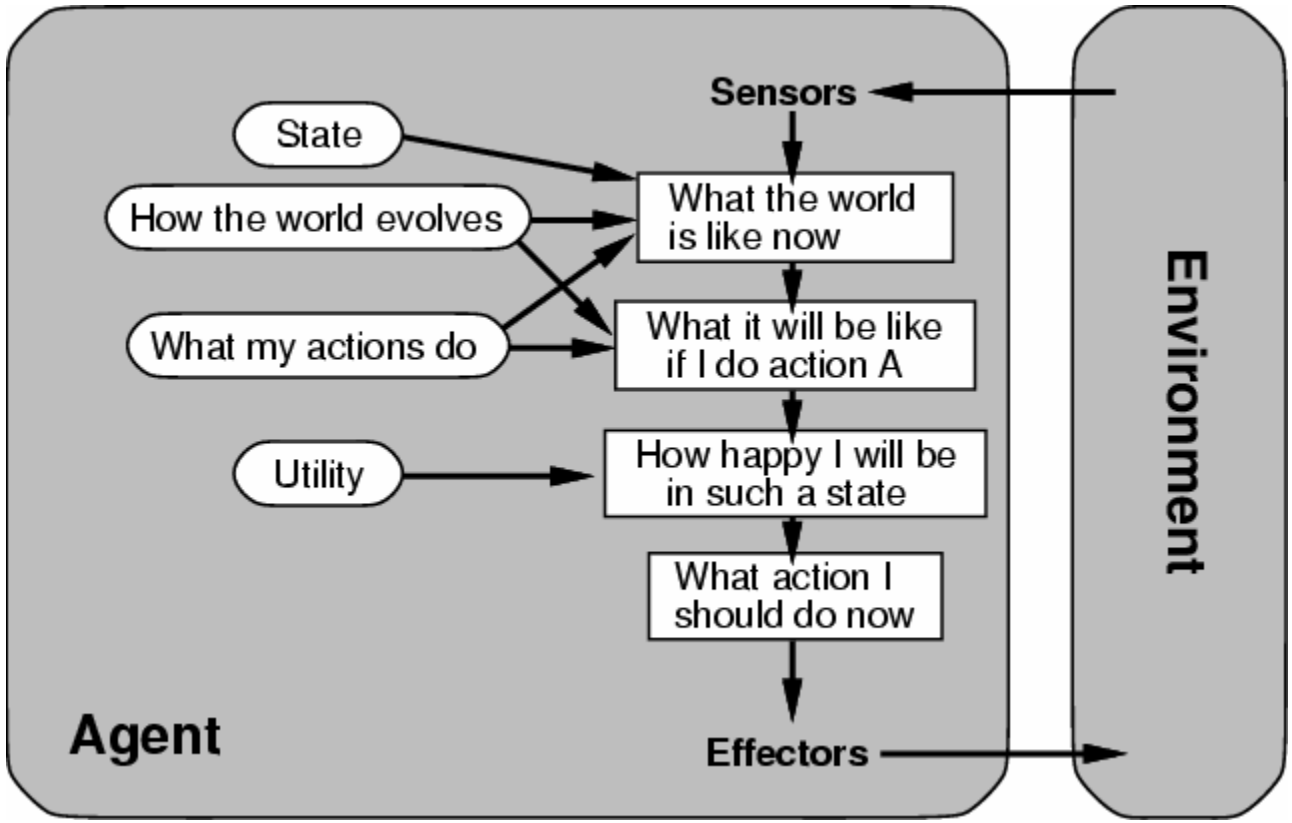
Major Agent Types
(4) Agents with Goals (cont)

- Knowing current state is **not always enough**.
- **State** allows an agent to **keep track of unseen parts** of the world, but the agent must **update state based on knowledge of changes** in the world and of effects of own actions.
- **Goal = description of desired situation**
- Example: (1) Decision to **change lanes** depends on a goal to go somewhere (and other factors); (2) **shopping well** depends on a shopping list, map of store, knowledge of menu

Major Agent Types
(4) Agents with Goals (cont)

- **Search** (Russell Chapters 3-5) and **Planning** (Chapters 11-13) are concerned with **finding sequences of actions to satisfy a goal.**
- **Reflexive agent** concerned with **one action at a time**.
- Planning as classically defined is finding a **sequence of actions** that achieves a goal.
- **Contrast with condition-action rules**: involves consideration of future "what will happen if I do ..." (fundamental difference).

Major Agent Types
(5) Utility-based Agents



Major Agent Types
(5) Utility-based Agents (cont)

- Preferred world state has **higher utility** for agent = quality of being useful
- Example: (1) quicker, safer, more reliable ways to get where going; (2) price comparison shopping
- Utility function: state->U(state) = measure of happiness
- Kinds of decisions allows:
  - choice between **conflicting goals** and
  - choice between **likelihood of success** and **importance of goal** (if achievement uncertain).
- Search (goal-based) vs. games (utilities).

Shopping Example Activities

- **Navigating**: Move around store; avoid obstacles
  - **Reflex agent**: store map compiled in.
  - **Goal-based agent**: create an internal map, reason explicitly about it, use signs and adapt to changes (e.g., specials at the ends of aisles).
- **Gathering**. Find and put into cart groceries it wants, need to induce objects from percepts.
  - **Reflex agent**: wander and grab items that look good.
  - **Goal-based agent**: shopping list.

Shopping Example Activities (cont)

- **Menu-planning**. Generate shopping list, modify list if store is out of some item.
  - **Goal-based agent**: required; what happens when a needed item is not there? Achieve the goal some other way. e.g., no milk cartons: get canned milk or powdered milk.

- **Choosing among alternative brands**
  - **utility-based agent**: trade off quality for price.

Agent Environments

- **Accessible/Inaccessible**: If an agent's sensors give it access to the **complete state of the environment** needed to choose an action, the environment is accessible to the agent. Such environments are convenient, since the agent is freed from the task of keeping track of the changes in the environment.

- **Deterministic/Nondeterministic**: An environment is deterministic for an agent if the **next state** of the environment is completely **determined by the current state** of the environment and the action of the agent. In an accessible and deterministic environment the agent need not deal with uncertainty.

- **Episodic/Nonepisodic**: An episodic environment means that **subsequent episodes do not depend on what actions occurred in previous episodes**. Such environments do not require the agent to plan ahead.

Agent Environments (cont)

- **Static/Dynamic**: **An environment which does not change** while the agent is thinking is static. In a static environment the agent need not worry about the passage of time while he is thinking, nor does he have to observe the world while he is thinking. In static environments the time it takes to compute a good strategy does not matter.

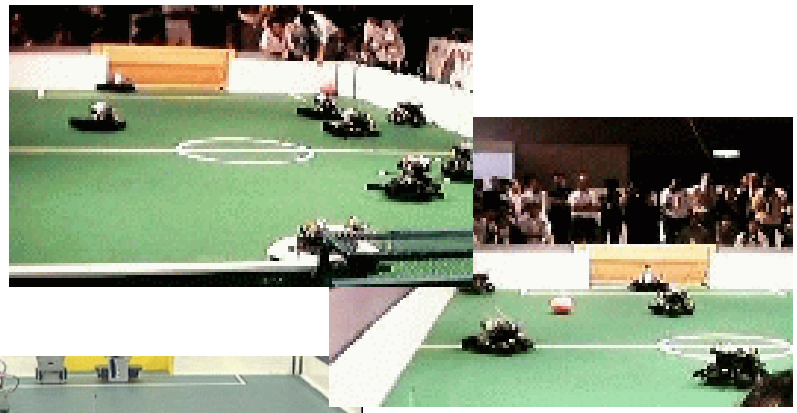
- **Discrete/Continuous**: If the number of **distinct percepts and actions is limited the environment is discrete**, otherwise it is continuous.

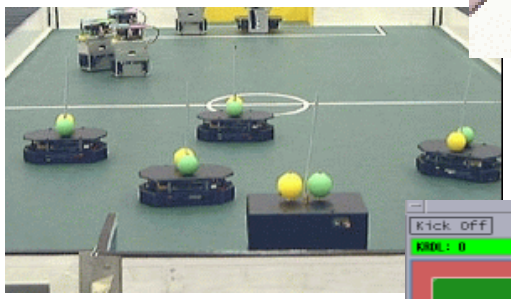
Characteristics of Some Environments

Environment	Accessible	Deterministic	Episodic	Static	Discrete
Chess with a clock	Yes	Yes	No	Semi	Yes
Chess without a clock	Yes	Yes	No	Yes	Yes
Poker	No	No	No	Yes	Yes
Backgammon	Yes	No	No	Yes	Yes
Taxi driving	No	No	No	No	No
Medical diagnosis system	No	No	No	No	No
Image-analysis system	Yes	Yes	Yes	Semi	No
Part-picking robot	No	No	Yes	No	No
Refinery controller	No	No	No	No	No
Interactive English tutor	No	No	No	No	Yes

Class Activity 2: To write the characteristics of the environment in Robocup domain

- Accessible/Inaccessible?
- Deterministic/Nondeterministic?
- Episodic/Nonepisodic?
- Static/Dynamic?
- Discrete/Continuous?





Basic Environment Simulator Program

```
procedure RUN-ENVIRONMENT(state, UPDATE-FN, agents, termination)
  inputs: state, the initial state of the environment
          UPDATE-FN, function to modify the environment
          agents, a set of agents
          termination, a predicate to test when we are done

  repeat
    for each agent in agents do
      PERCEPT[agent] ← GET-PERCEPT(agent, state)
    end
    for each agent in agents do
      ACTION[agent] ← PROGRAM[agent](PERCEPT[agent])
    end
    state ← UPDATE-FN(actions, agents, state)
  until termination(state)
```

- Each agent is **given its percept**, **gets an action** from each agent, and then **updates the environments** (eg. Robocup Soccer Server)

Environment Simulator Program with Performance Measurement

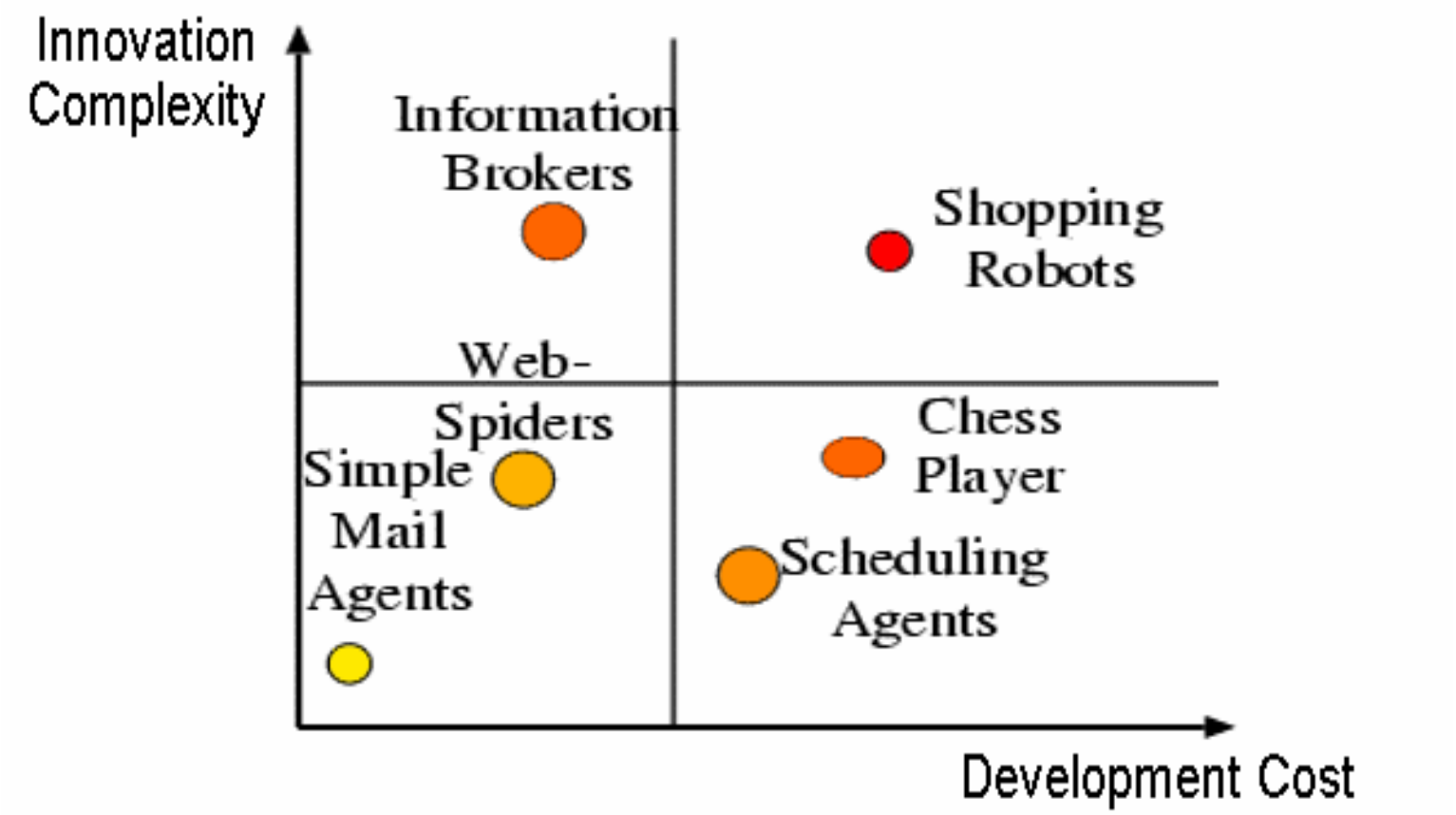
```
function RUN-EVAL-ENVIRONMENT(state, UPDATE-FN, agents,
                              termination, PERFORMANCE-FN) returns scores
  local variables: scores, a vector the same size as agents, all 0

  repeat
    for each agent in agents do
      PERCEPT[agent] ← GET-PERCEPT(agent, state)
    end
    for each agent in agents do
      ACTION[agent] ← PROGRAM[agent](PERCEPT[agent])
    end
    state ← UPDATE-FN(actions, agents, state)
    scores ← PERFORMANCE-FN(scores, agents, state)
  until termination(state)
  return scores                                          /* change */
```

- An environment simulator program that keep tracks of the **performance measure** for each agent (eg. SimCity)

Internet Environments for Agents

- Mail
- Newsgroups
- WWW
- FTP
- MUDs
- Internet Games

An Agent Portfolio

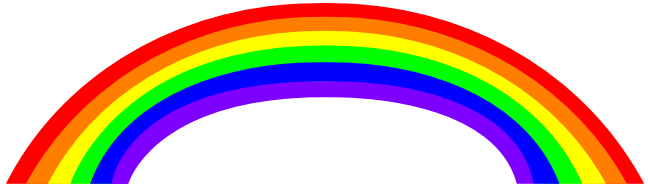


Connection to Search and Knowledge Representation

- **Goal-based** and **utility-based** agents require representation of:
  - world state
  - actions and effects
  - utilities

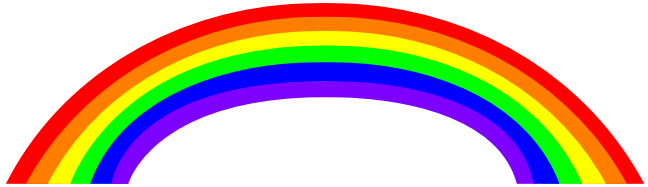
- Agents must be able to **reason with this knowledge**

Conclusion



- An agent **perceives** and **acts** in an environment. It has an **architecture** and is implemented by a **program**.
- An ideal agent always chooses the action which **maximizes its expected performance**, given the percept sequence received so far.
- An **autonomous agent** uses its **own experience** rather than built-in knowledge of the environment by the designer.
- An agent program maps from a **percept to an action** and **updates** its internal state.

Conclusion (cont)



- **Reflex** agents **respond immediately** to percepts.
- **Goal-based agents** act in order to **achieve their goal(s).**
- **Utility-based agents** maximize their own **utility function**.
- **Representing knowledge** is important for successful agent design.
- Some environments are more difficult for agents than others. The **most challenging** environments are **inaccessible**, **non-deterministic**, **non-episodic**, **dynamic**, and **continuous**.
  eg. the Robocup Challenge (Kitano, 1997)

Class Activity 3: “Intelligent Agents - The Right Information at the Right Time”

IBM Intelligent Agent White Paper http://www.networking.ibm.com/iag/iagwp1.html

Intelligent Agents:

The Right Information at the Right Time

Don Gilbert

http://www.networking.ibm.com/iag/iaghome.html

IBM Corporation
Research Triangle Park, NC USA
May, 1997

ABSTRACT

Intelligent agents are an emerging technology that is making computer systems easier to use by allowing people to delegate work back to the computer. They help do things like find and filter information, customize views of information, and automate work. This paper examines some real agent-enhanced applications to explore the value of agents, summarizes the characteristics that differentiate agents from other software, and lists IBM intelligent agent technologies which can be used to add agents to new or existing applications.

What's in Store for Lecture 2b

- Multi-Agent Systems
- Applications of Intelligent Agents
- Roadmap for Agent Research and Development
- Class Activity 2: Two papers on Multi-Agent System - Reading.

What's in Store for Lecture 3-5

- Formulation of search problems
- Uninformed (blind) Search Algorithms
- Informed (heuristic) Search Algorithms
- Game Playing
- Students' Mini Research Presentation by Group A
- Class Activity 1: One paper on Search Strategies - Reading

End of Lecture 2

Good Night.