

Personnel

Lecture 1

Artificial Intelligence (0270) 2002

In this first lecture of the course, we will have a general overview of the course, and a few essential ideas about Artificial Intelligence in general.

References

- Chapter 1: Section 1.1, 1.4
- Chapter 2: The whole chapter

Lecturer: Dr. Isaac To Kar Keung (iktko@csis.hku.hk), CYC407.

Office hour:

Tutors:

- Bai Zhenlong (z1bai@csis.hku.hk), CYC 321
Consultation hour:
- Chen Lin (lchen2@csis.hku.hk), CYC 319
Consultation hour:

Web site: <http://www.csis.hku.hk/~c0270/>

Text:

- “*Artificial Intelligence, A Modern Approach*”. Russell and Norvig, Prentice Hall. Web site:

<http://www.cs.berkeley.edu/~russell/aima.html>

AI(0270)

AI(0270)-1.1

Calling for help

If you have difficulties, you can do one of the followings:

- **Preferred:** post an article to our newsgroup, hku.csis.CSIS0270.
This allows all of you to answer and to benefit from the answer.

Questions asked in any other means may be answered in newsgroup.
We assume that you keep watching the newsgroup.
- Ask by **E-mail**. You can send to any of us, but it is better to just send mails to c0270@csis.hku.hk.
In this way every of us will receive your query.
- Contact us during our **office hours**.
- Make an **appointment** with one of us, and ask us face-to-face.
Send a mail to one of us before visiting my office to make an appointment.

AI(0270)-1.2

Lecture conduct

- **Talking** is allowed **only on course matters**.
- **Attend lectures** only if you are willing to learn.
Nobody force you into lectures anyway.
- **Everything** about the lecture **needs to be understood**.
Preferably during the lecture, and absolutely no later than the end of the day.
- **Ask** whenever you find something you don't understand.
Don't hesitate: what you don't understand is probably not understood by many others as well, and you're helping others by just asking.
- **Stop the lecture** if you need time to digest. This is **vitaly** important: it is usually the only way for me to know that you need more time.
This is the way in which you can understand everything during the lecture.

AI(0270)-1.3

Assessment

Assignments (35%)

- 2 programming assignments, approx. 20%.
Language: you can choose either C++ or Java.
Platform: one of Linux, our departmental servers, or Sun Java development platform.
- 3 written assignments, approx. 15%.

Quiz (15%), To be announced

- 1 hour quiz of easy questions. Should be very easy if you have done the assignments.
Big hint! Quiz is on assignments, not on other materials of the course.

Exam (50%)

AI(0270)-1.4

Advices on learning

- **Learn the concepts first (most important)**. Implementation details are not as important as the concepts.
We usually show code for illustration purpose. But if you are not familiar with computer code, it is okay to skim them.
- **Understand, not just memorize**. Find the underlying ideas that give rise to concepts. Find unifying themes.
- **Keep up with lectures**. Each lecture builds on top of the previous ones, so if you miss just one lecture, it will be significantly more difficult to catch up with upcoming lectures.
- **Do the assignments early**. Programming assignments are usually more difficult than what you first think. And assignments clear up your concepts for easy understanding of the coming lectures.
- **Read the book**. Lecture is too short to explain everything in the book, and the function of lectures is simply to guide you to read.

AI(0270)-1.5

What is Artificial Intelligence?

AI is the study of **making intelligent systems**.

Note the word "making": we will write programs.

But what is intelligent? Different approaches:

Think like human (cognitive modelling) Automate human thinking, e.g., decision making, problem solving, learning.	Think rationally (laws of thought) Perform computations that makes it possible to perceive, reason and act.
Act like human (Turing test) Do things that requires intelligence when performed by people.	Act rationally (rational agent) Explain and emulate intelligent behaviour in terms of computational process.

Note that we didn't specify the problem domain. This vagueness is due to the fact that we want **general strategies**, not specific to particular problems.

AI(0270)-1.6

Our approach

We adopt the "**Rational agent**" approach:

- Require **rational** behaviour: avoid the unclear idea about which behaviour is more "human-like". In contrast, what is rational is more well-defined (what is best given the current knowledge).
- Require **actions**, not just thinking: avoid the philosophical question about whether machines can think. It is also more general: we need the knowledges advocated by "thinking" camp in order to "act" rationally.

Combining the two advantages, we can ask whether an agent is better or worse than another in a scientific way.

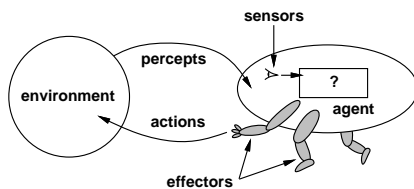
What is rational? Always do the right thing?

Unluckily, it is usually not possible, and we have to make compromises. E.g., it might be computationally impossible, or it might takes too much time or memory of the computer.

AI(0270)-1.7

Agent

- The book uses a term **agent** to mean anything that:
 - **perceives** the environment through sensors
 - and **act** through effectors.
- The primary aim of AI is to **make good agents**.
Expect to write some programs in this course, although not a whole lot of them.



AI(0270)-1.8

Rational agents

- We aim at **rational** agents, i.e., one which *does the right thing*.
- What is the right thing? The action that would make us **successful**. But successful in what?
- We need some **performance measures**: how to judge the system.
Must be fixed in advance, and need to be accurate.
- **When** to evaluate performance? We will need some averaging.

I see my old friend across the street, and want to chat with him. Noting that there is no traffic, and I've nothing else to do, I started crossing the street... only to get flattened in the middle of the road, by a cargo door falling off from a 747. Am I rational?

Come on... we don't have a crystal ball!

AI(0270)-1.9

What is rational

Rationality: **expected** success **given what has been perceived**.

What is rational is determined by

- The **performance measure**.
- What is **perceived** by the agent **so far**.
- What is known by the agent about the **environment**.
I won't anticipate a falling cargo door: the environment doesn't work that way.
- What **actions** the agent can do.
But remember that finding more information is a possible action!

An **ideal rational agent** will maximize the expected performance measure for **every possible percept sequence**, on the basis of the percepts and its built-in knowledge about the environment.

But remember that ideality is seldom possible in CS, especially in AI.

AI(0270)-1.10

PAGE descriptions

Thus before we create any AI agents (in fact, any programs), we must have a good idea about:

- What are the possible **percepts** and **actions**?
- What is the **performance measure**?
- What we know about the **environment**?

These are collectively called the PAGE description (Percept, Action, Goal, Environment) of the problem.

The environment might be real or artificial, and it doesn't make any difference at all. Some real environments are dull, while some artificial environments can be very difficult.

AI(0270)-1.11

Example

Agent Type	Percepts	Actions	Goals	Environment
Medical diagnosis system	Symptoms, findings, patient's answers	Questions, tests, treatments	Healthy patient, minimize costs	Patient, hospital
Satellite image analysis system	Pixels of varying intensity, color	Print a categorization of scene	Correct categorization	Images from orbiting satellite
Part-picking robot	Pixels of varying intensity	Pick up parts and sort into bins	Place parts in correct bins	Conveyor belt with parts
Refinery controller	Temperature, pressure readings	Open, close valves; adjust temperature	Maximize purity, yield, safety	Refinery
Interactive English tutor	Typed words	Print exercises, suggestions, corrections	Maximize student's score on test	Set of students

AI(0270)-1.12

Let's look it up!

So input is some percept, and output is some action. Thus our agent can be very simple.

```
map<vector<Percept>, Action> table; // To be initialized
vector<Percept> percepts;
Action TableDrivenAgent(Percept p) {
    percepts.push_back(p);
    return table[percepts];
}
```

We use a C++-like syntax here. But nearly everything here is not valid C++. You'll have to make things concrete when you implement your own programs.

- Note that we don't need to put the environment and the evaluation functions here. These information are encoded into the table.

For those who don't know C++ well enough, a vector is just an extensible array, a map is like a vector except the indices can be any comparable type rather than just positive integers, and the push_back operation of vector adds one object at the end of the array.

AI(0270)-1.13

The failure of lookup

The problem is, for most situations, **the above program won't work**.

- The **table** will be **huge**, spending a lot of memory.
E.g., to play "reverse it", the table will need 60! entries.
- It will take a **long time** for the designer to **fill the table entries**.
Both because the table is large, and because it requires some intelligence.
- If the environment or the evaluation function **changes** a little bit, **the table needs to be completely redesigned**.
In other words, the degree of automation is not high.
- It is very **difficult** for the program to **learn** by itself, because it needs to learn all entries of the huge table.
We want a smaller set of values to learn.

Note that "The program is not intelligent" is not a proper critic. The problem is that **the program is not satisfactory in certain areas**.

AI(0270)-1.14

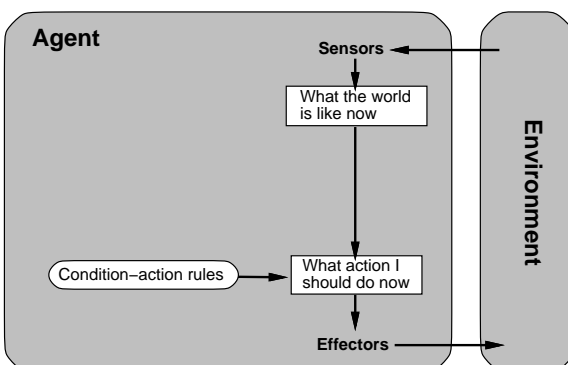
Reflex agent: making lookup more reasonable

If lookup is infeasible, what can we do to improve the situation?

- One possibility: seek **if-then rules** that govern, in the following form:
if car-in-front-is-braking then initiate-braking
- Such agents are called **reflex agent**: like reflex in neural-science, it is encoded into the program, and effected without reasoning.
- Example: to play a game of tic-tac-toe using table, we would need $3^9 = 19683$ entries.
- E.g., to play a game of tic-tac-toe, around 80 rules suffice.
If you write more complicated rules, 5 might suffice.
- But even in such simple situation, it is usually not a good idea to use reflex agents.
E.g., in TTT, point 2–4 are still quite true, and we can do better.

AI(0270)-1.15

A schematics of a reflex agent



AI(0270)-1.16

How to do better?

So what can we do to improve the situation?

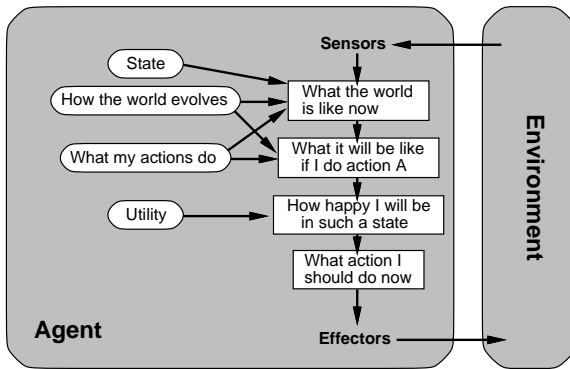
- Write code that **predicts the results** of a possible action.
- Explicit code the **evaluation function** into our program.
- Write code to **plan action sequences** to maximize performance.
- Find the **intrinsic properties** that can easily change, and write code that **learn** the values of these properties automatically.
Unluckily, the course does not have enough time to cover this.

E.g.: instead of writing the 80 rules governing the best playing strategy of TTT, we write code to play the game, given the specification of the game.

As a result, the program can play any similar game that requires the same level of expertise.

AI(0270)-1.17

A schematics of a utility-based agent



AI(0270)-1.18

A better agent

It is instructional to see what is needed by such a program.

Problem specific information hard-coded into the program

Generic functions that can be reused by similar problems

- How to get from percepts
- What are the possible action
- What is the result of actions
- What is considered winning
- How to put actions into effect
- Look for possible actions
- Predict what opponent will do
- Find the expected result
- Find the expected performance
- Find the best strategy

How difficult is it? Indeed, not that difficult.

So easy that it is not usually regarded as AI. You can find the program presented here (100 lines not counting interface code) in the source of this lecture.

AI(0270)-1.19

About the environment

There are a few things special about the environment that makes TTT particularly easy:

- It is **accessible**, i.e., everything about the environment can be perceived by the percepts. There is nothing "hidden away" from it.
- It is **deterministic**, i.e., the result of every action is completely fixed, i.e., nothing by chance.
- It is **discrete**, and has only a **small number** of possible percept and action values, i.e., we don't have a large number or continuous spectrum.
- It is **static**, i.e., the agent can spend any amount of time on the problem without the state of the environment changes by a little bit.
But this is not really very important: the problem is really simple.

Not every AI problem is this simple. Confronting more difficult AI problems, the programmer and the computer need to pay more effort.

AI(0270)-1.20

What's ahead

- Our primary business in this course is to learn **how reasoning takes place** in the computer.
- The tic-tac-toe example shows one of them: **by searching** for ways to succeed. We will learn how to search efficiently.
- Searching will be combined with a method to **represent knowledge** of the environment: logic.
- We will see how new knowledge can be generated from old ones, and how to combine it with searching to plan actions.
- Finally, we will learn how reasoning has to be done when there is uncertainty in our knowledge, which is common in **expert systems**.

What you should do now: get the book, and skim through the first two chapters.

AI(0270)-1.21

Course overview

- Wk 1: Introduction, basic AI searching (WA1)
- Wk 2: Informed search, constraint satisfaction problems
- Wk 3: Two-person Game (PA1)
- Wk 4 (after lunar holiday): Logic
- Wk 5: Planning (PA2)
- Wk 6: Logical reasoning
- Wk 7 (after reading week): (Quiz,) Chaining systems
- Wk 8: Resolution (WA2)
- Wk 9: (Tutorial only)
- Wk 10: Expert systems
- Wk 11: Belief network (WA3)
- Wk 12: Decision making
- Wk 13: Revision

AI(0270)-1.22