Rational Agents

Overview -> Rational Agents

Preparation

Reading Russel & Norvig Chapter (1-)2

If you find the text overwhelming, please focus on the keywords listed under Review of the reading assignment below.

Learning outcomes

After the completion of this workshop, students should

- be familiar with CodinGame and have successfully solved some of the recommended Medium puzzles.
- be able to analyse problems with respect to PEAS (performance measure, environment, actuator, sensor).
- understand and be able to explain what kind of agent they have implemented (e.g., reflex agent, planning agent, etc.) and whether their agents (or more generally, solution methods) are rational.
- be familiar on an introductory level with basic data structures and algorithms (e.g., lists, sets, trees, binary search, greedy search).
- be able to understand and answer the sample exam questions posted on Blackboard that relate to the material covered in this workshop.

Briefing

Recap from last week

We discuss the content from R&N Ch. 1 and the material studied last week. Two questions, state the most important point only:

1. What have you learnt so far?
2. What is your greatest challenge for today?

Keywords from last week

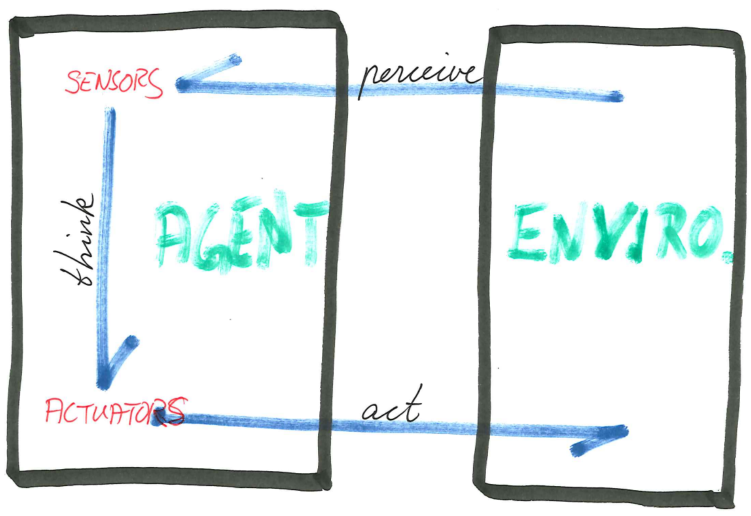

Figure 1: Agent is the perceive-think-act cycle

- four definitions of AI along the axes of thinking and acting humanly and rationally.
- foundations of AI, e.g., philosophy, mathematics, economics, neuroscience, psychology, computer engineering, control theory and cybernetics, linguistics
- the history of AI
- applications of AI and state-of-the-art

Todau: The Rational Agent

- Rational Agent: perceive and act
  - vs machine learning
- Choose the right action
  - what is right?
  - Hume on is-ought
  - Predefined goal
- Aristotle's algorithm implemented by Newell & Simon (GPL)
  - means-ends analysis
  - we deliberate on means, not on ends

Review of the reading assignment

We divide the group in three groups, to discuss each of the following topics from the reading assignments. Each topic should be discussed using the coding problems from last week as examples.

1. PEAS - Performace Measure, Environment, Actuators, Sensors
2. Properties of the task environment
   - Fully or partially observable
   - Single or multi-agent
   - Deterministic or not
   - Episodic or sequential
   - Static or dynamic
   - Discrete or continuous
   - Known or unknown
3. Program Type
   - simple reflex agent
   - model-based agents
   - goal-based agents
   - utility-based agents
   - learning agents

There are lecture notes prepared, but not necessarily used. See Lecture on Rational Agents

Exercise (AIMA Ch 2 on PEAS)

Working individually or in pairs, we do Exercise 5-6 (rephrased below) from the AIMA Exercises Ch. 2.

Consider one of the following problems (activities):

- (Group 1) Playing soccer.
- (Group 2) Exploring the subsurface oceans of Titan.
- (Group 3) Shopping for used AI books on the Internet.
- (Group 4) Playing a tennis match.
- (Group 5) Practicing tennis against a wall.
- (Group 6) Performing a high jump.
- (Group 7) Knitting a sweater.
- (Group 8) Bidding on an item at an auction.

For your activity,

1. Give a PEAS description of the task environment, i.e. characterise
   - **P**erformance Measure
   - **E**nvironment
   - **A**ctuators
   - **S**ensors
2. Characterise the problem in terms of the properties listed in Section 2.3.2.

3. Characterise possible agent types according to the properties listed in Section 2.4

If you have time to do two activities, please do the next one in the list, wrapping around from 8 to 1.

Programming Problems (CodinGame)

Preliminaries

Knowledge about various data structures is very useful, but it is also a big topic to explore. How do you store a large collection of data points?

In python, we very often use a list.

```
l = [ "Andrew", "Bella", "Charlie", "Denise" ]
l.append( "Eric" )    # new element at the (right hand) end
first = l.pop( 0 )    # remove from the head (left hand end)
last = l.pop( )       # rmove from the end (right hand end)
```

Lists are not very fast when they get large. If you look for a particular element, you need to search through all of them. The `pop(0)` call above is also likely to be slow, because it leaves a hole in the list and elements may have to be reshuffled. But as long as you your code is fast *enough*, you have better things to think about.

If you need to store elements with a lookup key, you can use a dict (dictionary)

```
d = { "foo": 12, "bar": 17 }
print(d["bar"])
d["foobar"] = 25
print(d)
```

In Java, the Collections Framework includes various useful interfaces and classes, e.g. + Set: HashSet, TreeSet, LinkedHashSet + List: ArrayList, LinkedList + Deque: ArrayDeque, LinkedList + Map: HashMap, TreeMap, LinkedHashMap

In Python there are also libraries, but not as systematically organised. For instance, there is a Queue class.

Important Please ask if you need/want to look into this. I do not want to use a lot of time to discuss these preliminaries, because for some of you, it is undergraduate material. However, I am happy to discuss problems with those of you who need it. If these data structures are new to you, you should spend some time on the algorithmic problems below, and ask for help.

Intelligent Agents

The following problesm are intended to demonstrate and challenge the intelligent agent view introduced thus far. Three rules to keep in mind.

1. For each challenge you solve, discuss how it the problem and your solution can be classified using the terminology of R&N Ch. 2.

2. Remember that you make **autonomous** robots.
    - An important part of the problem is to imagine what the robot will know at any given point in the program, and reason how to act in the given situation.
3. Use **pair programming**.
    - This is a well tested and proven technique for programming and problem solving.
    - If you think you prefer to work alone, you still have to **try** pair programming this once, and then you can tell me afterwards. Thus, *team up in pairs* and choose a problem to attack.

Suggested problems

1. The Descent
    - *This was also given in Week 1.*
    - Description: The enterprise is in danger: drawn towards the surface of an unknown planet, it is at risk of crashing against towering mountains. Help Kirk and Spock destroy the mountains… Save the enterprise!
    - Topic: Search in an array
2. Shadows of the Knight Episode 1 (medium): intervals, binary search
    - Intelligent Agent
    - Some similarities with the Þor problem last week, but this time the target location is not precisely known
3. Mars Lander — Episode 1
    - *This was also given in Week 1.*
    - Description: You have been promoted to commander of the Mars Lander mission! The goal of the operation is to land an exploration rover on martian ground. Your superiors at NASA expect very much of you for this mission, and you'll have to prove that you have what it takes to become a great intersideral commander. You will have to land the space ship on mars, making sure that the landing is done smoothly.
    - Topic: Speed regulation
4. The Fall E1
    - Here you have to predict a robot's movement, rather than direct it.
5. Blunder Episode 1
    - Description: Bender is a depressed robot who heals his depression by partying and drinking alcohol.
    - To save him from a life of debauchery, his creators have reprogrammed the control system with a more rudimentary intelligence.
    - Unfortunately, he has lost his sense of humor and his former friends have now rejected him.
    - Bender is now all alone and is wandering through the streets of

    Futurama with the intention of ending it all in a suicide booth.
    - To intercept him and save him from almost certain death, the authorities have given you a mission: write a program that will make it possible to foresee the path that Bender follows. To do so, you are given the logic for the new intelligence with which Bender has been programmed as well as a map of the city.
6. Bender Episode 4
    - This is a challenge. You have to take care to build the model and the search tree.
    - Note that you have to find the entire path in your model, and then execute your program. You cannot use a perception loop to try and err.
    - Work in **groups**. Discuss the model and make sure you can agree on an understanding.
    - Previous episodes no longer available

Feel free to look for additional challenges to solve.

Algorithmic Challenges

The following challenges are more basic algorithmic problems, which may be usefual to practice programming and problem solving in general.
Do not worry if you do not have time for them; they are intended for those who need a further and a different challenge.

- There is no Spoon (medium): lists
    - Search problem
    - Using lists is not the only solution
- Telephone Numbers (medium): trees, sets
    - Calculation - modelling challenge
    - many solutions - can be simulated
- Advanced tree (medium): tree
    - Build and print a tree
- The Gift (medium): greedy algorithms
    - Mathematical problem

Lunch-time Heads Up

We review the experiences made in CodinGame.

- How do you classify the problems you have worked on in the PEAS framework? (Consider the questions from the briefing exercise (5-6 AIMA Ch 2.)
- What does the PEAS framework tell us about how to program a solutions?

Debrief

Module evaluation

- Status report.
- Questions and Answers?

Exercise 1 (AIMA Ch. 2)

This exercise we should as a plenary seminar.

> Suppose that the performance measure is concerned with just the first T time steps of the environment and ignores everything thereafter. Show that a rational agent's action may depend not just on the state of the environment but also on the time step it has reached.

Homework

Prepare yourself for the next workshop by doing the following:

1. Review what you have learnt this session.
2. Do the preparation for the next lecture, on Search.