Lecture Notes on Intelligent Agents Spring 2013

From the course by full prof. Stefania Costantini, University of L'Aquila

Federico Gobbo

Lecture Notes on Intelligent Agents Spring 2013

From the course by full prof. Stefania Costantini, University of L'Aquila

Federico Gobbo

This book is for sale at http://leanpub.com/LectureNotesIntelligentAgentsSpring2013

This version was published on 2013-06-12

Tweet This Book!

Please help Federico Gobbo by spreading the word about this book on Twitter!

The suggested hashtag for this book is #AI2AtAQ2013.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search/#AI2AtAQ2013

to the city of L'Aquila

Contents

Presentation of the course

The textbook by David Poole, Alan Mackworth and Randy Goebel: Computational Intelligence, A Logical Approach[1] is the place where to start.

Here, we will explore the **weak AI** perspective – i.e., we are *not* interested into consciousness in the machine, but rather 'only' in the *behaviours* that appear to be intelligent. Perhaps the interaction with the environment will let consciousness emerge, but this question is let to the philosophers, such as the assessment of risk derived from the application of AI in real-world situations (e.g., CSER[2]).

In the book *the Society of Mind* – which is also an open course at the MIT[3] – Marvin Minsky deals a lot with **common sense reasoning**. We want to formalise our everyday practices in logical terms. Of course, we should **go beyond classic logic**, and we want to implement it into the machine: this is the *ratio* behind Computational Logic. For example, expert systems can be a valid auxiliary tools for diagnosis. Expert systems are not agent-based, because an expert system is a single block which can do reasoning on its knowledge base. Like software agents, they are disembodied. Another difference is that they do have neither body nor emotions – in principle, in AI you can insert emotions, for example see Furby[4]. Marvin Minsky deals with this problem in his book too.

Also take a look at freely available **ontologies**, such as OpenCyc[5] for common sense reasoning, or Wordnet[6] for natural language semantics, or even Senso Comune[7], which is specific for the Italian language.

We take the **Prolog language as granted**, and we are always in contact with the group at the Imperial College in AI, whose mentor is Robert Kowalski. Of course, **neural networks** can be used **to complement** our symbolic perspective resulting in a hybrid approach.

Our main topic here is the following one: **software intelligent agents**. First, they are **autonomus**: if you don't 'kill' them, they stay alive. Second, they interact through **events** in an environment, that is perceived at some degree. How to widen our perception? One of the goals is exactly to expand the range of their perceptions. The most basic intelligent agent is the *thermostat*: now, we try to transform it into a smart object, that for example could learn from the user's preferences. Third, they have **goals**. For example, take care of the budget learnt from the user of the thermostat.

There is a theory that starts from the idea that intelligence is not needed: **reactive agents**. You need only to model the action-reaction reflex and the intelligence emerges from the environment (something like Unix daemons). The pioneer is **Rodney Brooks** – see his web page at the MIT[8]. They succeeded to model insects, pets, and other artificial lifes, embodied into robots. This is a very important research area: the sense of RoboCup[9] (robots playing football) is to study the strategies of teamwork – think about a team of robots which help people after an earthquake.

We prefer to explore **deliberative agents**, without dealing with problems of embodiement – no robots here, by the moment. They have **beliefs**, **desires** and **intentions**. We will formalise in Computational

[1] http://www.cs.ubc.ca/~poole/ci.html
[2] http://cser.org/
[3] http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-868j-the-society-of-mind-spring-2007/index.htm
[4] http://en.wikipedia.org/wiki/Furby
[5] http://www.cyc.com/platform/opencyc/
[6] http://wordnet.princeton.edu/
[7] http://www.sensocomune.it/
[8] http://people.csail.mit.edu/brooks/
[9] http://www.robocup.org/

Logic terms these notions. Deliberative agents are **proactive**: basing theirselves on the past events and their knowledge in general, it can decide to act into the world, for instance to send messages to the other agents (social interaction). They can be described in terms of **plans**.

The main difference between an object and the agent is this one: the behaviour of the objects is predictable, in the case of agents this does not apply. So, the **object-oriented paradigm** and the agent-oriented paradigm are different. Objects are instances of classes, which is a collection of features and behaviours, so it's easy to clone them. For example, the class of cars can have very different models. It is described in terms of know-hows, that are methods which are invoked on request. The original idea by Adele Goldberg was the inspiration by real-world objects. The same method can give you a polymorphic result, for example the method 'accelerate' is different if you have a Cinquecento or a Ferrari.

In the **agent-oriented paradigm** you are not the concept of inheritance, unlike objects (we are talking in general). Agents do not act for free: they 'want' to do things according for example to the goals. Agents are conceptually different threads of control, and so they can be concurrent. They do not invoke methods, they send messages and in order to perform actions. They can be used for a lot of different purposes.

Since *1956*, agents were used to do **symbolic manipulation** through the reasoner, **like theorem provers** (deductive reasoning agents). Unlike theorem provers, agents interact with the environment. *Preconditions* are expressed in logical formulas: for example, to open a door the precondition is that the door should not be already open! Sometimes my beliefs don't match reality (the door is locked), and I have to change my plan – this is modeled straightforwardly within the negation-as-failure, because we assume not to be Gods and hence our knowledge is not complete by definition. This family of agents is precise but it is terribly slow. If the environment changes quickier than my capacity of reasoning I will be always late. The problem is the complexity of the selction function. It is known in the literature as the problem of **brittleness**.

Platforms for building autonomous software require concepts embedded into **first-class objects**: observations, events, actions, beliefs, and then also messages, goals, etc.

Do we need all of them? Theoretically, it suffices to use a Turing-complete language to implement everything. But practically this is not feasible. For example, let's take the lists in LISP and Prolog, which is native; convercely in C you have to build it from scratch, and so you should reason at another level. So, we should use agent-oriented programming languages. You can also adapt Java to the agent-oriented paradigm, but you should write the inferential engine!

How many **paradigms of intelligent agents exist**? A lot! We, here at L'Aquila, have **DALI**, which was invented by Stefania Costantini, implemented by Arianna Tocchio – and yes, the name is origined by Daniele and Alice, the names of Costantini's children. We will delve into it. It is general-purpose with respect to the approaches within the agent-oriented paradigm. Other languages are more keen to specific approaches.

The first seminal work on agents is dated in 1977, by Carl Hewitt[10]'s actor model starts in comparison with objects: "a self-contained, interactive and concurrently-executing object" which was called 'actor' – see the seminal paper[11] here. On the other side of the story, recently agents were put into a network, and what you obtain Multi-Agents Systems (MAS).

Let's see how we can design an agent architecture. Different architectures reflect differen points of view. The simplest architecture is this one:

E is the set of environment states

[10] http://carlhewitt.info/

[11] http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-194.pdf

Ac is the set of agent possible actions `E --> Ac`

This is the purely reactive agent. But something is missing: where is the connection with the environment? What is missing is the *sensors*. How we can complete it? Events become perceptions, which modify the internal state and therefore determines the next action, which becomes an event. The chain is defined by Kowalski as *observe, think, act*. The next lesson, we will start to see **practical reasoning**, the paradigm of agents in use since 1990.