Artificial Intelligence I: introduction

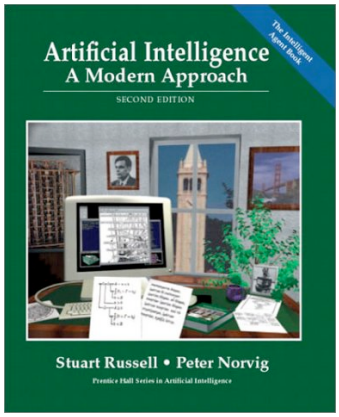
Lecturer: Tom Lenaerts

Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle (IRIDIA)

Université Libre de Bruxelles

Practical stuff

- Course homepage:
  - **http://iridia.ulb.ac.be/~tlenaert/teach/ai1.html**
- Mailinglist:
  - **02452-valvas@elvas.vub.ac.be**
- Textbook:
  - **S. Russell and P. Norvig *Artificial Intelligence:A Modern Approach* Prentice Hall, 2003, Second Edition**
- Excercices:
  - **Joachim de Beule en Bart De Vylder.**
  - **Lisp (~scheme++)**
- Exam: written at end of 1st. sem.



Course overview

- What is AI.
- Intelligent agents.
- Problem solving.
- Knowledge and reasoning.
- Planning.
- Uncertain knowledge and reasoning.
- Learning.
- Communicating, perceiving and acting.

Outline

- What is AI
- A brief history
- The State of the art (see book)
- Reading
- Lisp
  - **And its relation to Scheme**

What is Artificial Intelligence

- Creative extension of philosophy:
  - **Understand and BUILD intelligent entities**
- Origin after WWII
- Highly interdisciplinary
- Currently consist of huge variety of subfields
  - **This course will discuss some of them**

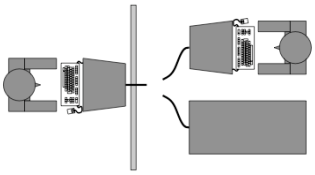
What is Artificial Intelligence

- Different definitions due to different criteria
  - **Two dimensions:**
    - Thought processes/reasoning vs. behavior/action
    - Success according to human standards vs. success according to an ideal concept of intelligence: rationality.

Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that act rationally

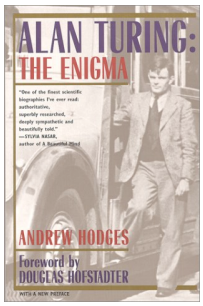
Systems that act like humans

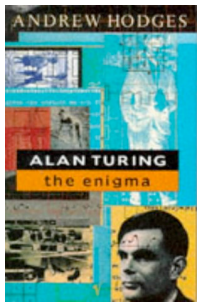
- When does a system behave intelligently?
  - **Turing (1950) *Computing Machinery and Intelligence***
  - **Operational test of intelligence: imitation game**



  - **Test still relevant now, yet might be the wrong question.**
  - **Requires the collaboration of major components of AI: knowledge, reasoning, language understanding, learning, ...**

Systems that act like humans





Andrew Hodges.
Alan Turing, the enigma
Available at amazon.co.uk

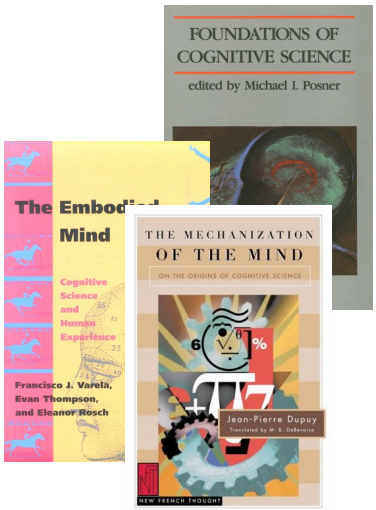
Problem with Turing test: not reproducible, constructive or amenable to mathematical analysis.

Systems that think like humans

- How do humans think?
  - **Requires scientific theories of internal brain activities (cognitive model):**
    - Level of abstraction? (knowledge or circuitry?)
    - Validation?
      - **Predicting and testing human behavior**
      - **Identification from neurological data**
  - **Cognitive Science vs. Cognitive neuroscience.**
- Both approaches are now distinct from AI
- Share that the available theories do not explain anything resembling human intelligence.
  - **Three fields share a principal direction.**

Systems that think like humans

- Some references;
  - **Daniel C. Dennet. Consciousness explained.**
  - **M. Posner (edt.) Foundations of cognitive science**
  - **Francisco J. Varela et al. The Embodied Mind**
  - **J.-P. Dupuy. The mechanization of the mind**

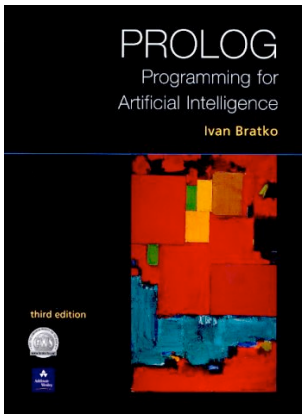


Systems that think rationally

- Capturing the laws of thought
  - **Aristotle: What are 'correct' argument and thought processes?**
    - Correctness depends on irrefutability of reasoning processes.
  - **This study initiated the field of logic.**
    - The logicist tradition in AI hopes to create intelligent systems using logic programming.
  - **Problems:**
    - Not all intelligence is mediated by logic behavior
    - What is the purpose of thinking? What thought should one have?

Systems that think rationally

- A reference;
  - **Ivan Bratko, Prolog programming for artificial intelligence.**

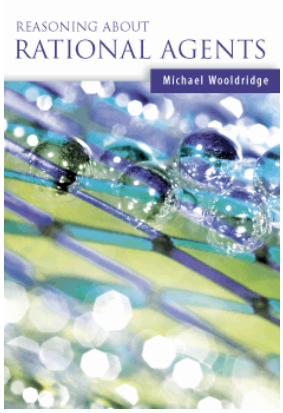


Systems that act rationally

- Rational behavior: "doing the right thing"
- The "Right thing" is that what is expected to *maximize goal achievement given the available information*.
- Can include thinking, yet in service of rational action.
  - **Action without thinking: e.g. reflexes.**
- Two advantages over previous approaches:
  - **More general than law of thoughts approach**
  - **More amenable to scientific development.**
- Yet rationality is only applicable in *ideal* environments.
- Moreover rationality is not a very good model of reality.

Systems that act rationally

- Some references;
  - **Michael Wooldridge. Reasoning about rational agents.**



Rational agents

- An agent is an entity that perceives and acts
- This course is about designing rational agents
  - **An agent is a function from percept histories to actions:**

$$f : P^* \rightarrow A$$

  - **For any given class of environments and task we seek the agent (or class of agents) with the best performance.**
  - **Problem: computational limitations make perfect rationality unachievable.**

Foundations of AI

- Different fields have contributed to AI in the form of ideas,viewpoints and techniques.
  - **Philosophy: Logic, reasoning, mind as a physical system, foundations of learning, language and rationality.**
  - **Mathematics: Formal representation and proof algorithms, computation, (un)decidability, (in)tractability, probability.**
  - **Psychology: adaptation, phenomena of perception and motor control.**
  - **Economics: formal theory of rational decisions, game theory.**
  - **Linguistics: knowledge represetatio, grammar.**
  - **Neuroscience: physical substrate for mental activities.**
  - **Control theory: homeostatic systems, stability, optimal agent design.**

A brief history

- What happened after WWII?
  - **1943: Warren Mc Culloch and Walter Pitts: a model of artificial boolean neurons to perform computations.**
    - First steps toward connectionist computation and learning (Hebbian learning).
    - Marvin Minsky and Dann Edmonds (1951) constructed the first neural network computer
  - **1950: Alan Turing's "Computing Machinery and Intelligence"**
    - First complete vision of AI.

A brief history (2)

- The birth of AI (1956)
  - **Darmouth Workshop bringing together top minds on automata theory, neural nets and the study of intelligence.**
    - Allen Newell and Herbert Simon: The logic theorist (first nonnumerical thinking program used for theorem proving)
    - For the next 20 years the field was dominated by these participants.
  - **Great expectations (1952-1969)**
    - Newell and Simon introduced the General Problem Solver.
      - **Imitation of human problem-solving**
    - Arthur Samuel (1952-)investigated game playing (checkers) with great success.
    - John McCarthy(1958-) :
      - **Inventor of Lisp (second-oldest high-level language)**
      - **Logic oriented, Advice Taker (separation between knowledge and reasoning)**

A brief history (3)

- The birth of AI (1956)
  - **Great expectations continued ..**
    - Marvin Minsky (1958 -)
      - **Introduction of microworlds that appear to require intelligence to solve: e.g. blocks-world.**
      - **Anti-logic orientation, society of the mind.**
- Collapse in AI research (1966 - 1973)
  - **Progress was slower than expected.**
    - Unrealistic predictions.
  - **Some systems lacked scalability.**
    - Combinatorial explosion in search.
  - **Fundamental limitations on techniques and representations.**
    - Minsky and Papert (1969) Perceptrons.

A brief history (4)

- AI revival through knowledge-based systems (1969-1970)
  - **General-purpose vs. domain specific**
    - E.g. the DENDRAL project (Buchanan et al. 1969)
      - **First successful knowledge intensive system.**
  - **Expert systems**
    - MYCIN to diagnose blood infections (Feigenbaum et al.)
      - **Introduction of uncertainty in reasoning.**
  - **Increase in knowledge representation research.**
    - Logic, frames, semantic nets, …

A brief history (5)

- AI becomes an industry (1980 - present)
  - **R1 at DEC (McDermott, 1982)**
  - **Fifth generation project in Japan (1981)**
  - **American response …**
- Puts an end to the AI winter.
- Connectionist revival (1986 - present)
  - **Parallel distributed processing (RumelHart and McClelland, 1986); backprop.**

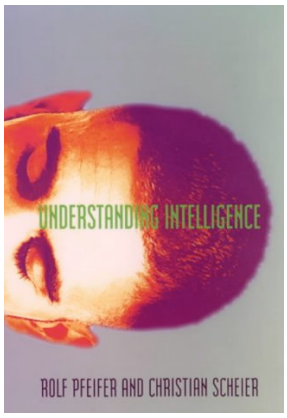
A brief history (6)

- AI becomes a science (1987 - present)
  - **Neats vs. scruffies.**
    - In speech recognition: hidden markov models
    - In neural networks
    - In uncertain reasoning and expert systems: Bayesian network formalism
    - …
- The emergence of intelligent agents (1995 - present)
  - **The whole agent problem:**

    "How does an agent act/behave embedded in real environments with continuous sensory inputs"

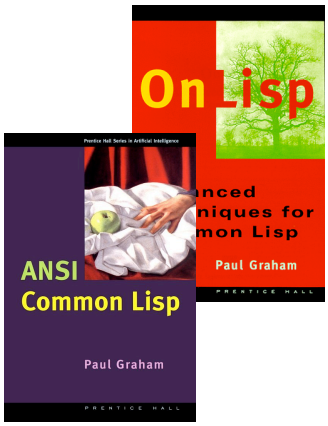
Reading

- One course on AI provides a limited view on a vast research area
- Reading assignment:
  - **P.1-138**
  - **Provides background information on the current perspective in AI: *embodied cognitive science*.**



Lisp vs. scheme

- Lisp (= LISt Processor) is the second oldest programming language still in use (after FORTRAN).
- Invented by John McCarthy at MIT in 1958.
- Until the mid '80s Lisp was more a family of dialects than a single language.
- In 1986 an ANSI subcommittee was formed to standardize these dialects into a single Common Lisp.
  - The result being the first Object Oriented language to become standardized, in 1994.
- Once you understand Common Lisp it is easy to adapt yourself to weaker dialects as for instance Scheme.



Lisp vs Scheme

- *Lisp* has much more built-in functions and special forms, the *Scheme* language definition takes 45 pages while Common Lisp takes 1029 pages)
- Apart from lexical variables(lexically scoped) *Lisp* also has special variables (dynamically scoped)
  - **In a lexically scoped language, the scope of an identifier is fixed at compile time to some region in the source code containing the identifier's declaration. This means that an identifier is only accessible within that region (including procedures declared within it).**
  - **In a dynamically scoped language an identifier can be referred to, not only in the block where it is declared, but also in any function or procedure called from within that block, even if the called procedure is declared outside the block**

Lisp vs Scheme

- Statically vs dynamically scoped variables

```
>(set 'regular 5)
5
>(defun check-regular ()
        regular)
CHECK-REGULAR
>(check-regular)
5
> (let ((regular 6))
    (check-regular))
5
```

```
>(defvar *special* 5)
*SPECIAL*

>(defun check-special ()
        *special*)
CHECK-SPECIAL
>(check-special)
5
>(let ((*special* 6))
        (check-special))
6
```

Lisp vs Scheme

- *Scheme* evaluates the function part of a function call in exactly the same way as arguments, *Lisp* doesn't.
  - **In Lisp, the role of the symbol depends on the position in the list**

```
(fun arg)
```

    - Example:

```
(let ((list '(1 2 3)))
      (list list))
==>((1 2 3))
```

  - **Function calls: Scheme vs. Lisp**

```
(let ((fun (compute-a-function)))
        (fun x y)

(map car L)
```

```
let ((fun (compute-a-function)))
        (funcall fun x y))

(map 'list #'car L)
```


Lisp vs Scheme

- *Scheme* uses one name space for functions, variables, etc., *Lisp* doesn't.
  - ***Special (global) symbols***

    (defun square (x) …)
    (setf (symbol-function square) (x) …)

  - ***Lexical (local) symbols***

    (labels ((square (x) …)) …)
    (setf square (function (lambda (x) …))

Lisp vs Scheme

- In *Lisp* defun defines functions in the global environment even if the function is defined internally to another function.

```
(define (stack)
  (let ((data '()))
    (define (push elm) …)
    (define (pop) …)
    …)
```

```
(defun stack ()
  (let ((data '()))
    (defun push (elm) …)
    (defun pop () …)
    …)
```

```
(defun stack ()
  (let ((data '()))
    (labels ((push (elm) …)
             (pop () …))
      …)
```


Lisp vs Scheme

- *Lisp* functions can have rest and optional parameters. *Scheme* functions only can have the equivalent of a rest parameter.

```
((lambda (a b) (+ a (* b 3))) 4 5) => 19
((lambda (a &optional (b 2)) (+ a (* b 3))) 4 5) => 19
((lambda (a &optional (b 2)) (+ a (* b 3))) 4) => 10
((lambda (&optional (a 2 b) (c 3 d) &rest x) (list a b c d x)))
        => (2 nil 3 nil nil)
((lambda (&optional (a 2 b) (c 3 d) &rest x) (list a b c d x))  6)
        => (6 t 3 nil nil)
```

Lisp vs Scheme

- *Lisp* functions can have also keyword parameters.

```
((lambda (a b &key c d) (list a b c d)) 1 2)    => (1 2 nil nil)
((lambda (a b &key c d) (list a b c d)) 1 2 :c 6)    => (1 2 6 nil)
((lambda (a b &key c d) (list a b c d)) 1 2 :d 8)    => (1 2 nil 8)
((lambda (a b &key c d) (list a b c d)) 1 2 :c 6 :d 8)    => (1 2 6 8)
((lambda (a b &key c d) (list a b c d)) 1 2 :d 8 :c 6)    => (1 2 6 8)
```

- Or some mixture.

```
((lambda (a &optional (b 3) &rest x &key c (d a))
         (list a b c d x))  1)   => (1 3 nil 1 ())
((lambda (a &optional (b 3) &rest x &key c (d a))
         (list a b c d x))  1 2)   => (1 2 nil 1 ())
((lambda (a &optional (b 3) &rest x &key c (d a))
         (list a b c d x))  :c 7)   => (:c 7 nil :c ())
```


Lisp vs Scheme

- *Lisp* has standard macros, *Scheme* since R5RS.
- *"Lisp macros are a way to execute arbitrary code at "compile time", using entities that are called like functions, but evaluate their arguments (or not, if they choose not to) in ways that are controlled by the macro itself. The language used to write the macro is just Lisp itself, so the full power of the language is available"*
- Allows you to define your own special forms as `if` or `and`.

Lisp vs Scheme

- *Lisp* has special forms (*loop*, *do*, *dotimes*, …) for looping, in *Scheme* the user is asked to use tail-recursion that is implemented efficiently.

```
(loop for i fixnum from 3
      when (oddp i) collect i
      while (< i 5))
⇒   (3 5)
```

http://iridia.ulb.ac.be/~tlenaert/prog/Lisp/cltl/cltl2.html

Other courses at the VUB

- AI does not end here …
  - **Artificiele Intelligentie II**
  - **Technieken van de AI I en II**
  - **Autonomous Agents**
  - **Adaptive Systems I en II**
  - **Machine Learning**
  - **Multi-agent systems**
  - **…**

Some references

- Understanding Intelligence by Rolf Pfeifer and Christian Scheier.
- Artificial Intelligence: Structures and Strategies for Complex Problem-solving by George Luger.
- Computation and Intelligence: Collective readings edited by George Luger.
- Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp by Peter Norvig.