Knowledge representation and reasoning Lecture 1: Introduction

Natasha Alechina

nza@cs.nott.ac.uk

・ 同 ト ・ ヨ ト ・ ヨ ト

Plan of the lecture



- 2 What is this module about
- 3 Famous knowledge-based systems
- 4 Plan of the module

・ロト ・ 日 ・ ・ 日 ・ ・ 日 ・

3

Module summary

- 2 lectures a week
- 100 % exam
- Exam papers (and answers) on the web
- webpage: http://www.cs.nott.ac.uk/~nza/G53KRR
- textbook:

Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning.* Elsevier, 2004

my web page has a link to Levesque's lecture slides; I will be mostly using a board, so prepare to take notes!

ロマネロマネルボマネー

What is this module about

- How can knowledge be represented symbolically and manipulated in an automated way by reasoning programs
- Knowledge: some information about the world
 - medical information about some particular set of diseases: what causes them, how to diagnose them
 - geographical data: which city is the capital of which country, population statistics, ...
 - common sense physics: bodies cannot go through solid walls, ...
- Representation: how / in which language do we represent this information
- Reasoning: how to extract more information from what is explicitly represented (because we cannot represent every single fact explicitly as in a database)

・ コット (雪) ・ ヨット (雪) ・ ヨ

Knowledge-based systems

- We want to be able to talk about some AI programs in terms of what they 'know'
 - (which corresponds to taking 'intentional stance' towards those systems, ascribing them human characteristics - for why this may be useful, see Daniel Dennett)
- and not just talk about what they know but also have something to point to in those systems corresponding to 'knowledge' and determining their behaviour, namely *explicitly represented symbolic knowledge*

ヘロト ヘ戸ト ヘヨト ヘヨト

Example (Brachman and Levesque)

Two Prolog programs with identical behaviour: printColour(snow) :- !, write("It's white."). printColour(grass) :- !, write("It's green."). printColour(sky) :- !, write("It's yellow."). printColour(X) :- !, write("Beats me.").

and

```
printColour(X) :- colour(X,Y), !, write("It's "),
write(Y), write(".").
printColour(X) :- write("Beats me.").
colour(snow, white).
colour(sky, yellow).
colour(X,Z) : - madeof(X,Z), colour(Z,Y).
madeof(grass, vegetation).
colour(vegetation, green).
```

Which one is knowledge-based

- Only the second program has explicit representation of 'knowledge' that snow is white
- the second program does what it does when asked for the colour of snow because of this knowledge. When colour (snow, white) is removed, it will not print the right colour for snow.
- what makes the system knowledge-based is not
 - the use of a particular logical-looking language like Prolog
 - or having representation of true facts (colour (sky, yellow) is not)
 - or having lots of facts, or having a complex structure
- rather, it is having explicit representation of knowledge which is used in the operation of the program

Definition of knowledge-based systems and knowledge bases

- Knowledge-based systems are systems for which intentional stance is grounded by design in symbolic representation
- The symbolic representation of knowledge is called a knowledge base.

How to do representation and reasoning: one possible answer

- Representation: as a set of sentences of first order logic
- Reasoning: deducing logical consequences
- We will see later in the course that
 - other languages may be more convenient and efficient for some applications
 - sometimes we want to produce good guesses based on available information as well as proper logical consequences

ロトス得たくほとくほと

Examples of knowledge-based systems

Various expert systems

- MYCIN (1970s, Stanford University)
- XCON (1978, Carnegie Mellon University)
- Perhaps most famous knowledge base: CYC (1980s, Douglas Lenat, Cycorp, Austin, Texas)
- Modern examples: ontologies
 - Snomed CT http://snomed.dataline.co.uk/
 - Gene ontology http://www.geneontology.org/

ロトスポトメラトメラト

MYCIN

- 1970s, Stanford University (Edward Shortliffe, Pat Buchanan)
- Production rule system (we will see them later in the course)
- Purpose: automatic diagnosis of bacterial infections
- Lots of interviews with experts on infectious diseases, translated into rules (knowledge acquisition is a non-trivial process; also see later in the course)
- approximately 500 rules

イロト 不得 トイヨト イヨト

Example MYCIN rule

```
Rule in LISP:

RULE035

PREMISE: ($ AND (SAME CNTXT GRAM GRAMNEG)

(SAME CNTXT MORPH ROD)

(SAME CNTXT AIR ANAEROBIC))

ACTION: (CONCLUDE CNTXT IDENTITY BACTEROIDES TALLY

.6)

English translation:

IF:
```

- 1 the gram stain of the organism is gramneg, and
- 2 the morphology of the organism is rod, and
- 3 the aerobicity of the organism is anaerobic

THEN: There is suggestive evidence (.6) that the identity of the organism is bacteroides

・ 同 ト ・ ヨ ト ・ ヨ ト

3

More about MYCIN

- some facts and some conclusions of the rules (as above) are not absolutely certain
- MYCIN uses numerical *certainty factors*; range between -1 and 1
- (reasonably involved) rules for combining certainty factors of premises, with the number in the rule (as 0.6 above) into a certainty factor for the conclusions
- Iater it turned out that MYCIN's recommendations would have been the same if it used only 4 values for certainty factors
- MYCIN was never used in practice (ethical and legal issues)
- when tested on real cases, did as well or better than the members of the Stanford medical school

XCON

- John McDermott, CMU, 1978
- eXpert CONfigurer system for configuring VAX computers
- production rule system, written using OPS5 (language for production systems, implemented in LISP)
- 10,000 rules
- used commercially

Сус

- The Cyc Knowledge Server is a very large knowledge base and inference engine
- Developed by Cycorp: http://www.cyc.com/
- It aims to provide a deep layer of 'common sense knowledge', to be used by other knowledge-intensive programs

Cyc knowledge base

- Contains terms and assertions in formal language CycL, based first-order logic, syntax similar to LISP
- Knowledge base contains classification of things (starting with the most general category: Thing), and also facts, rules of thumb, heuristics for reasoning about everyday objects
- Currently, over 200,000 terms, and many human-entered assertions involving each term; Cyc can derive new assertions from those
- Divided in thousands of 'microtheories'

Cyc knowledge base

- General knowledge: things, intangible things, physical objects, individuals, collections, sets, relations...
- Domain-specific knowledge, for example:
 - Political geography: general information (e.g. What is a border?) and specific information about towns, cities, countries and international organizations
 - Human anatomy and physiology
 - Chemistry
 - Iots of others see Cycorp web page

Snomed

- Snomed CT: Systematized Nomenclature of Medicine Clinical Terms
- Developed by College of American Pathologists and NHS
- Clinical terminology (with formal definitions)
- Designed for unambiguous recording of data and interoperability with software applications
- Uses ontology language (different from first order logic) EL++
- Approx. 400 000 concepts, 1 million terms and 1.6 million relationships

・ロト ・ 同ト ・ ヨト ・ ヨト ・ ヨ

Snomed: example

Concept: 32553006 - Hangover Descriptions: Synonym: hangover effect Synonym: hangover from alcohol Relationships: (is a) 228273003 - Finding relating to alcohol drinking behavior (causative agent) 311492009 - Ingestible alcohol

Plan of the module

- First order logic (3 lectures)
- Expressing knowledge (1 lecture)
- Resolution (4 lectures)
- Horn clauses, backward chaining, forward chaining
- Rules in production systems
- How to build a knowledge based system
- Description logic
- Defaults/non-monotonic reasoning
- Uncertainty/bayesian networks

・ 同 ト ・ ヨ ト ・ ヨ ト ・

Recommended reading for the next lecture

Brachman and Levesque, chapter 2 (The language of first-order logic).

・ ロ ト ス 厚 ト ス ヨ ト ス ヨ ト