Comp24412: Symbolic AI
Lecture 0: Course Introduction

Ian Pratt-Hartmann

Room KB2.38: email: ipratt@cs.man.ac.uk

2016–17
• In 2011, the question-answering system Watson, developed by IBM, won a special round of the TV quiz programme Jeopardy!:

![Image of Jeopardy! with contestants and question screens]

• The program uses a variety of text-processing and information-retrieval techniques to produce answers to familiar quiz-type questions, such as

> Kathleen Kenyon’s excavation of this city mentioned in Joshua showed the walls had been repaired 17 times.

• Despite its sophistication, the program has little idea of what it is doing.
One aspect of understanding that Watson lacks is an appreciation of logical relationships expressed in language.

Here are some valid arguments:

Every beekeeper is an artist
Fred is a beekeeper
Fred is an artist

Every beekeeper is an artist
Some beekeepers are not carpenters
Some artists are not carpenters

Here is an invalid argument

Every beekeeper is an artist
Some carpenters are not artists
Some beekeepers are not carpenters

Question: how do we distinguish between the two?
• The question seems first to have been addressed by Aristotle (384–322 B.C)

• Aristotle attempted to catalogue all the valid argument forms, or syllogisms

• This is all explained in the (utterly impenetrable) *Prior Analytics*
• The question seems first to have been addressed by Aristotle (384–322 B.C.)

• Aristotle attempted to catalogue all the valid argument forms, or syllogisms

• This is all explained in the (utterly impenetrable) *Prior Analytics*
• Aristotle had a rather restricted view of the forms of sentences occurring in arguments.

• Three nineteenth century gentlemen attempted to broaden the scope of this logic:

  Augustus De Morgan  George Boole  W.S. Jevons

• De Morgan complained that Aristotle could not account for the evident validity of

  Every horse is an animal
  Every horse’s head is an animal’s head
Likewise, the syllogistic does not help us with

Every artist admires a beekeeper
Every beekeeper curses every artist who admires him
Every artist admires a beekeeper who curses him

Actually, de Morgan also looked in detail at adding numbers to syllogisms:

At least 13 artists are beekeepers
At most 3 beekeepers are carpenters
At most 4 dentists are not carpenters
At least 6 artists are not dentists.
Here is a real killer:

At most 1 artist admires at most 7 beekeepers
At most 2 carpenters admire at most 8 dentists
At most 3 artists admire at least 7 electricians
At most 4 beekeepers are not electricians
At most 5 dentists are not electricians
At most 1 beekeeper is a dentist
Here is a real killer:

At most 1 artist admires at most 7 beekeepers
At most 2 carpenters admire at most 8 dentists
At most 3 artists admire at least 7 electricians
At most 4 beekeepers are not electricians
At most 5 dentists are not electricians
At most 1 beekeeper is a dentist
At most 6 artists are carpenters

But you knew that, didn't you?
• Here is a real killer:

At most 1 artist admires at most 7 beekeepers
At most 2 carpenters admire at most 8 dentists
At most 3 artists admire at least 7 electricians
At most 4 beekeepers are not electricians
At most 5 dentists are not electricians
At most 1 beekeeper is a dentist
At most 6 artists are carpenters

• But you knew that, didn’t you?
• But the most decisive advance was made at the end of the nineteenth century by Gottlob Frege and Bertrand Russell.

• They developed **formal** languages to represent information:

Every artist admires a beekeeper
Every beekeeper curses every artist who admires him
Every artist admires a beekeeper who curses him

\[ \forall x (\text{artst}(x) \rightarrow \exists y (\text{bkpr}(y) \land \text{adm}(x, y))) \]
\[ \forall x (\text{bkpr}(x) \rightarrow \forall y (\text{artst}(y) \land \text{adm}(y, x) \rightarrow \text{crs}(x, y))) \]
\[ \forall x (\text{artst}(x) \rightarrow \exists y (\text{bkpr}(y) \land \text{crs}(y, x) \rightarrow \text{adm}(x, y))) \]
• Erm, actually, Frege’s notation, called the *Begriffsschrift*, was not quite so user-friendly:

But we won’t worry about it.
• In the 1980s, serious attempts were made to use logical methods in Artificial Intelligence.

• Here is one early robot, SRI’s Shakey:

• The key idea behind logical methods in AI can be summarized as follows
  • The agent’s knowledge is represented as a collection formulas in some logic
  • Reasoning tasks (planning, deduction, hypothesis formation, learning) is realized by the manipulation of these formulas.
• Perhaps the most ambitious knowledge-based AI project is CYC.
  • Started 1984 at MCC
  • Contains a huge knowledge-base codifying knowledge of everyday facts.
  • Retrieval of information is by logical inference.
  • Originally estimated that CYC would require 250,000 rules and 350 man-years.
  • Total effort to date is over a man-millennium . . .

• It is now widely accepted that this approach will not, by itself, succeed in producing intelligence.
• This course is about logical approaches to AI.
• We shall be concerned with the relationship between logic, natural language and deduction.
• Logic is not the whole of AI, but it is most probably part of it.
• Here is what we have to learn:
  1. Prolog programming
  2. First-order logic
  3. Applications of Logic in AI
  4. Natural language syntax
  5. Lambda calculus
  6. Natural language semantics
  7. Resolution theorem-proving
A student completing this course should:

1. have a working knowledge of the Prolog programming language
2. understand the fundamentals of natural language syntax
3. be able to compute the meanings of a range of natural language sentences
4. understand the operation and use of automated theorem-provers, and the theoretical reasons for their limitations
5. Understand how logic may be used to solve problems in AI.
and recommended reading:

- P. Blackburn and J. Bos, *Representation and Inference for Natural Language*, CSLI Press, 2005
- Almost any other Prolog book
# Lecture plan

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Lecture 1</th>
<th>Lecture 2</th>
<th>Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.01.17</td>
<td>L0: Introduction</td>
<td>L1: Prolog I</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>06.02.17</td>
<td>L2: Prolog II</td>
<td>L3: Prolog III</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>13.02.17</td>
<td>Prolog Catchup</td>
<td>L4: Search</td>
<td>Intro Prolog (D)</td>
</tr>
<tr>
<td>4</td>
<td>20.02.17</td>
<td>L5: Logic</td>
<td>L6: Logic and AI</td>
<td>N Queens (D)</td>
</tr>
<tr>
<td>5</td>
<td>27.02.17</td>
<td>L9: Grammar I</td>
<td>L10: Grammar II</td>
<td>Syntax in Prolog (D)</td>
</tr>
<tr>
<td>6</td>
<td>06.03.17</td>
<td>L9: Parsing I</td>
<td>L12: Parsing II</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>13.03.17</td>
<td>L13: Semantics I</td>
<td>L14: Semantics II</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>20.03.17</td>
<td>L15: Grammar III</td>
<td>L16: Semantics III</td>
<td>Semantics in Prolog (D)</td>
</tr>
<tr>
<td>9</td>
<td>27.03.17</td>
<td>L7: Logic and Prolog</td>
<td>L8: Resolution</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>24.04.17</td>
<td>L17: Finale</td>
<td>Catch-up</td>
<td>Reasoning in NL (D)</td>
</tr>
<tr>
<td>11</td>
<td>1.05.17</td>
<td>Revision</td>
<td>Revision</td>
<td>Marking</td>
</tr>
<tr>
<td>12</td>
<td>8.05.17</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Q: What do I do now?
A: Several things:

- Try out Prolog by typing `pl` to the teaching (Linux) system. (You should have `/opt/prolog/bin/` in your `$PATH`)
- Start reading *Learn Prolog Now!* (or any other book on Prolog). There are also various on-line introductions to Prolog, which are probably just as good.
- Buy a copy of *Representation and Inference for Natural Language*