Lisp and Prolog Background

Thank you to Prof. Roosen-Runge
for the slides on
background and history.

Lisp History

- Lisp--invented (**about 1958**) by the logician and AI researcher **John McCarthy**
- Widely used in for applications in AI every since
- Used in industry
  - » **To develop expert systems & other AI applications**
  - » **For Rapid Application Development (RAD)**

Lisp Use

- Lisp is used as an embedded language
- Found inside applications like Emacs (MLisp) and AutoCAD (AutoLisp).
  - » **Emacs supports Lisp as a language for developing programs**
  - » **As well as embedded language for controlling and customizing the editor itself**

Lisp availability

- On all major and many minor platforms.
- Numerous free- and shareware versions.
- Standard: Common Lisp

Prolog history

- Prolog invented (≈1972) by the AI researcher Alan Colmeraurer
  - » **Used at York in the Student Information System to check applications for input errors**
- Widely used to develop expert systems & other AI applications including natural language processing
  - » **Early ideas developed at University of Montreal; then University of Marseilles**

Prolog Use & Availability

- Prolog rumored to be embedded in MS Office
- On all major and many minor platforms
- Several free and shareware versions
- Standard: ‘Edinburgh-style’

Survival Value

McCarthy's Lisp

versus

Newell's IPL (Information Processing Language)

- **Both men pioneers in AI and computer science**
- **Both proposed languages at about the same time for symbolic computing to be used in AI research**

And the winner is ?

McCarthy?

Newell?

Why?

& the winner is ...

- Value of ‘high-level’ over ‘low-level’ language wasn’t so clearly recognized in the early days.
- Newell made what turned out to be a fatal error for his language

⇒ **Newell modeled IPL on the assembler language for an early computer instead of timeless logic**

→ **This made his language very operational**

→ **Couldn't be understood in mathematical terms.**

→ **Wasn't accepted as a notation for AI programs.**

Low- and High-level – 1

- Lisp became the ‘assembler language’ – lower-level – for AI programmers
  - » **Used to build higher-level systems**
  - » **Wilensky Chapters 21 & 22 give the basis for Prolog!**
- Common Lisp = union of the techniques and tools people have found useful.

Low- and High-level – 2

Prolog is a higher-level language for knowledge-based programming

- **More powerful, not necessarily as efficient**
- **More compact**
- **More understandable programs.**

Pure Lisp

- Denotational, functional rather than operational
  - » **No states: just a mapping**
    between arguments and result
    or, from input to output.
- ‘Pure’ Lisp is all we will have time to discuss in this course
  - » **It's what makes Lisp distinctive**

Pure Prolog

- ‘Pure’ Prolog: denotational & declarative
- Just 1 state
  - » **A ‘knowledge’ base = database for facts**
- This turned out to be a very big advance!

Lisp vs. Prolog?

Which AI language an AI researcher uses often depends on where they studied.

- **At MIT and Stanford, almost all Lisp**
- **MIT has used a dialect of Lisp called Scheme in their first year programming course for many years.**
- **At Edinburgh, almost all Prolog**

At York

- We have been more a ‘Prolog shop’ than a ‘Lisp shop’ in this department.

- Prof. Stachniak teaches a 4th year course on Logic Programming which includes a more advanced look at Prolog.

Other Choices

- There are many denotational, functional languages other than Lisp
  - » **ML is popular in British and some European universities**

    American universities tend to use Lisp
- No strong competitors to Prolog at present, mainly variants, extensions, and dialects.
- Objected-oriented add-ons available for both languages

Lisp in ‘Production’ Work

- Programmers tend to use operational features
  - **This is not good; it is the fault of poor software-development tools**
  - **There are claims that 90% of function calls in Lisp programs are to assignment functions – like setq and rplaca!**
- Assignment changes state and is sure sign of an operational description
  - **Compare the palindrome programs**