5. Special Purpose Languages

- The languages we have looked at so far, as well as the paradigms on which they are based, are intended as general purpose languages, i.e. we expect that every programming task can be done using the languages.

- There are languages that were developed with a special purpose in mind (also called domain specific languages).

- Some of these languages later were extended to become general purpose languages and usually this was done using one of the paradigms we looked at.
Special Purpose Languages (cont.)

- Usually, special purpose languages require more intelligence in compiler/interpreter and run-time system and one reason why some of these languages never were successful among the target audience were bad compilers/interpreters and run-time systems that essentially threw away any advantages of the languages.

- In this sense, Haskell or PROLOG come near to special purpose languages, although they were Turing-complete from the beginning.

- But special purposes definitely will also in future drive development of programming languages.
Examples

- Text processing/display:
  - TeX, LaTeX
  - HTML

- Data base creation/interaction
  - SQL

- Education/teaching
  - Pascal, Modula

- Symbolic Math
  - Mathematica
  - Matlab
Examples (cont.)

- Statistics
  - R
  - SPSS
- Hardware description languages
  - Verilog
  - VHDL
- Computer games
  - Maya
  - Unreal Engine
- Spreadsheets
  - Excel
Examples (cont.)

- Languages for multi-agent systems
  - Jade
  - netlogo

- Languages for embedded systems
  - Usually company-specific (and kept as private as possible)

- Systems for programming by demonstration/examples
  see the following
Special purpose language vs library

- Library “extends” already existing language and therefore requires being able to program in that language. Special purpose language can be easier to program in.
- Special purpose language might be too specific for a particular user application. Library is easier changed than language.
- Special purpose language only defines what “intelligence” need to be realized and different implementations are possible. Library is its own implementation.
Special purpose language vs library (cont.)

- Security: special purpose language run-time system code can be hidden ↔ library usually has to be open
5.1 Some history:
Special treatment of arrays: APL

- Arrays and especially matrixes are the basis of many mathematical and engineering programs and a language that offers a lot of special treatment for them should be of quite some interest

- APL (A Programming Language) was developed in the mid-1960s at IBM as a special purpose language for exactly this (although it could be used as general purpose language also)

- It included a large collection of operators on matrixes, but due to this APL programs were difficult to read and required in the beginning special terminals to write them on
Some history:
Text processing and SNOBOL (I)

- Manipulation of strings is an important task for text editors (and Natural Language Processing)
- While general purpose languages can do all necessary manipulations, it can require rather complex procedures to implement some of these manipulations
- SNOBOL was designed already incorporating a lot of these manipulations as build-in operators and it contained the concept of string pattern matching (which is pattern matching with the build-in knowledge about strings)
Some history:
Text processing and SNOBOL (II)

- String pattern matching is a more complex problem than "normal" pattern matching and consequently the resulting implementations of SNOBOL had serious performance issues.
- Operations that were "natural" in SNOBOL were finding certain substrings, testing that a word was a beginning word of a sentence or an end word and so on.
- The Unix awk command can be seen as a successor of SNOBOL, as can be more modern scripting languages.

CPSC 449 Principles of Programming Languages

Jörg Denzinger