CEng 242, Programming Language Concepts

Introduction

Onur Tolga Şehitoğlu

Computer Engineering

Introduction

- Church-Turing hypothesis say all programming languages and computational devices have the same power regarding computability.
- If you can define a computation in one of the universal programming languages, you can define the same computation in any other universal programming language.
- Why do we have so many programming languages?
- Is it desirable to know as much distinct P.L. as possible?

Course Objectives

- *Not* to teach specific programming languages. Haskell, C++ and Prolog languages are tools of this course.
- Studying the common concepts of programming languages.
- Studying the different paradigms and approaches.
- What is the measure of **quality** in a programming language?
- Construct a basis for other topics like compiler design, software engineering, object oriented design, human computer interaction...

Related Areas

- Human computer interaction
- Operating systems
- Computer Architecture
- Databases and information retrieval
- Software engineering

Programming Languages vs Natural Languages

- Formal vs Informal
- Strict rules of well-formedness vs Error tolerant
- Restricted vs Unrestricted domain

What makes a language programming language?

Is any formally defined language a programming language? (HTML?)

- **Universal:** All computation problems should be expressable.
  condition+(loop and/or recursion)
- **Natural:** All features required for the application domain.
  Fortran: numerical computation, COBOL: file processing, LISP tree and list operations.
- **Implementable:** It is possible to write a compiler or interpreter working on a computer.
  Mathematics, natural language?
- **Efficient:** Works with acceptable amount of CPU and memory.

What makes a language programming language?

Is any formally defined language a programming language? (HTML?)

- **Universal:** All computation problems should be expressable.
  condition+(loop and/or recursion)
- **Natural:** All features required for the application domain.
  Fortran: numerical computation, COBOL: file processing, LISP tree and list operations.
- **Implementable:** It is possible to write a compiler or interpreter working on a computer.
  Mathematics, natural language?
- **Efficient:** Works with acceptable amount of CPU and memory.

What makes a language programming language?

Is any formally defined language a programming language? (HTML?)

- **Universal:** All computation problems should be expressable. condition+(loop and/or recursion)
- **Natural:** All features required for the application domain. Fortran: numerical computation, COBOL: file processing, LISP tree and list operations.
- **Implementable:** It is possible to write a compiler or interpreter working on a computer.
  Mathematics, natural language?
- **Efficient:** Works with acceptable amount of CPU and memory.

What makes a language programming language?

Is any formally defined language a programming language? (HTML?)

- **Universal:** All computation problems should be expressable. condition+(loop and/or recursion)
- **Natural:** All features required for the application domain. Fortran: numerical computation, COBOL: file processing, LISP tree and list operations.
- **Implementable:** It is possible to write a compiler or interpreter working on a computer.
  Mathematics, natural language?
- **Efficient:** Works with acceptable amount of CPU and memory.

What makes a language programming language?

Is any formally defined language a programming language? (HTML?)

- **Universal:** All computation problems should be expressable. condition+(loop and/or recursion)
- **Natural:** All features required for the application domain. Fortran: numerical computation, COBOL: file processing, LISP tree and list operations.
- **Implementable:** It is possible to write a compiler or interpreter working on a computer.
  Mathematics, natural language?
- **Efficient:** Works with acceptable amount of CPU and memory.

"Hello world" in different languages:
`https://helloworldcollection.github.io/`

Paradigms

Paradigm: Theoretical or model frame. A model forming a basis for all similar approaches.

- **Imperative** Fortran, Cobol, Algol, Pascal, C, C++, Java, Basic.
- **Functional** Lisp, Scheme, ML, Haskell, Miranda
- **Object Oriented** Smalltalk, C++, Object Pascal, Eifell, Java, Csharp.
- **Concurrent** Ada, Occam, Par-C, Pict, Oz
- **Logic** Prolog, Icon
- **Constraint Programming** CLP, CHR
- **Mixed** Parallel Prolog, Oz (A functional, logic, object oriented, concurrent language: `http://www.mozart-oz.org`)

Syntax and Semantics

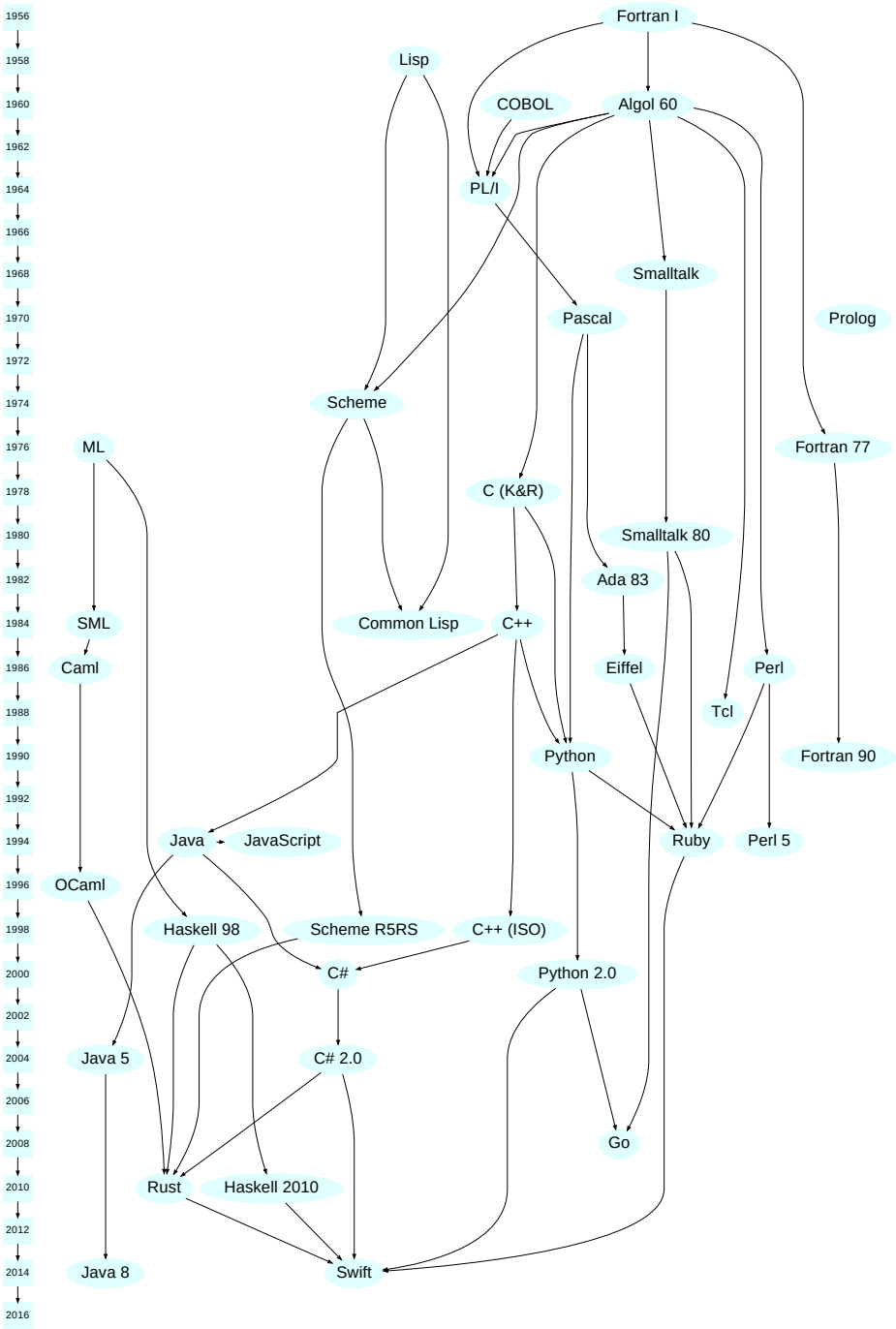
- **Syntax** Form. How language is structured, how it is expressed.
- Syntax is represented by Context Free Grammars expressed in BNF (Bacus Naur Form) notation (See CEng 280, Formal Languages and Abstract Machines)
- **Semantics** What does a program mean? How it works.

Syntax and Semantics

- **Syntax** Form. How language is structured, how it is expressed.
- Syntax is represented by Context Free Grammars expressed in BNF (Bacus Naur Form) notation (See CEng 280, Formal Languages and Abstract Machines)
- **Semantics** What does a program mean? How it works.

Language Processors

- Compilers (gcc, javac, f77)
- Interpreters (scheme, hugs, sml, bash)
- Beautifiers, pretty printers (a2ps, ident)
- Syntax directed editors (vim, anjuta, eclipse, visual studio)
- Validators (vgrind, lint)
- Verifiers



Taken from:
`http://rigaux.org/language-study/`

O'reiley poster:
http://archive.oreilly.com/pub/a/oreilly/news/languageposter_0504.html
download link