CSE 505

Lecture 1

Aug 27, 2012

Course Website

http://www.cse.buffalo.edu/LRG/CSE505
Language Research Group

Course Staff

Dr. Bharat Jayaraman, Professor
Office Hours: Tuesdays at 1-3 pm
338G Davis Hall
bharat@buffalo.edu

Bing Zhang, Teaching Assistant
Ph.D. student
Office Hours: TBA
bingzhan@buffalo.edu

Course Overview

- ➢ Procedural Languages
- ➢ Type Systems
- ➢ Object-Oriented Languages
- ➢ Execution/Runtime Issues
- ➢ Functional Languages
- ➢ Logic & Constraint Languages
- ➢ Domain-Specific Languages

Course Details

- Procedural Languages: scope rules, recursion, parameters, binding time, copying vs sharing, advanced control structures
- Type Systems: forms of polymorphism, strong vs weak typing, type safety, type inference, exceptions, abstract types, signatures, axioms
- Object-Oriented Languages: classes, objects, inheritance, polymorphism, design patterns
- Execution/Run-time Issues: compilation vs interpretation, reference counting vs garbage collection, abstract machines (JVM), debugging

Course Outline (cont'd)

- Functional Languages: lambda calculus, higher-order functions, rules and pattern-matching, lazy evaluation,
- Logic and Constraint Languages: rules, unification, search, grammars, negation, sets, constraints, meta-programming
- Domain-Specific Languages: very high-level languages motivated by specific applications, e.g., HTML, XML, SQL, YACC, etc.; scripting languages; visual interfaces

Prerequisites

- Recursive programming techniques
- Undergraduate-level data structures and discrete mathematics
- Knowledge of a modern object-oriented language, such as Java and C++

Course Materials

- Required Reading:
  Lecture Notes by B. Jayaraman,
  Great Lakes Printing,
  UB Commons
- On reserve in Sci & Eng Library:
  PL + language reference books

Assignments and Exams

- Assignments (40%)
- Mid-Term Exam (25%)
  - in-class, open-book
  - Mon, Oct 22, 2012
- Final Exam (35%)
  - Mon, Dec 10, 2012

Programming Projects

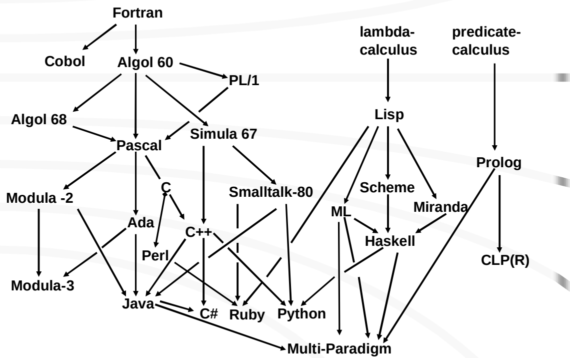
- Help deepen understanding of PL concepts. Not meant as “applications programming“
- On-line code submission.
- No late submissions. Ample time will be given.

Academic Integrity

- Oral discussion is fine.
- Written work must be your own, and not copied from another student.
- Do not copy verbatim from any source: text book, website, etc. Taking ideas from such sources is acceptable with proper citation.
- Violators may receive F grade on the entire course.

Imperative Languages

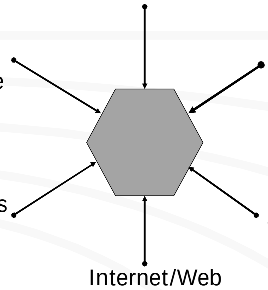
Declarative Languages



Language Families

- ➢ Programming Paradigms
  Procedural, Modular, Object-Oriented,
  Functional, Logic, Constraint, …
  Multi-Paradigm Languages
- ➢ Fundamental Issues
  Data (imperative) vs Control (declarative)
  Static (compiled) vs Dynamic (interpreted)
  Strongly Typed vs Weakly Typed
  Meta-level vs Object-level

Major Forces that shaped PLs



Mathematical

Computer Architecture

Software Engineering

Applications

Standardization

Internet/Web

Forces: Computer Architecture

- ➢ von Neumann computer architecture:
  - sequential control, updateable storage
- ➢ parallel computer architecture:
  - explicit parallelism
    process, fork/join, array-processing
  - implicit parallelism
    divide-conquer, search, dataflow,
- ➢ distributed/networked systems
  - distributed objects (CORBA, COM),
    remote method invocation

Forces: Software Engineering

- ➢ Information Hiding
  modules, interfaces, encapsulation
- ➢ Verification
  invariants, disciplined control
- ➢ Software Specification
  Unified Modeling Language

Forces: Mathematical

- ➢ Lambda Calculus
  LISP
- ➢ Predicate Calculus (Horn Clauses)
  Prolog, CLP®
- ➢ Set Theory
  SETL, SuRE

Forces: Applications

General PL concepts arising from special domains:

Simulation → Classes, Inheritance, Abstraction

Theorem-Proving → Polymorphic Types

Operating Systems → C

Windows and GUI applications → OOP

Forces: Internet/Web

- ➢ Applet, Servlet, …
  Java
- ➢ Scripting Languages
  Javascript, Ruby, Python

Forces: Standardization

- ➢ ANSI, ISO, DoD
  Cobol, C, C++, Ada, …

Contributions of individual PLs

FORTRAN – variables, arrays, expressions, simple control structures, subprograms

COBOL – first language standard, records/structs

Algol 60 – recursion, lexical scope, inner procedures

Algol 68 – advent of types, user-defined types, strong typing, flex arrays

Contributions of PLS (cont'd)

Simula 67 – class, inheritance, data abstraction

Pascal – popular type system, postfix code

C – HLL for writing operating systems (Unix)

Smalltalk (72—80) – OOP and GUIs

Modula-2 – module interface vs implementation

Contributions of PLS (cont'd)

ML– strong typing via type inference

C++ – strongly typed OOL, parametric (template) and subclass polymorphims

Java – banishes pointer variables, interfaces as types, applets, rich libraries for GUI-building

C# - many similarities with Java, but designed for the .NET framework

Contributions of PLS (cont'd)

Lisp – functional proramming, list processing, automatic storage management

Prolog – logic programming, pattern-matching, rule-based programming

CLP® - constraint logic programming over the reals ®, rules + constraints

Contributions of PLS (cont'd)

Perl – interpreted language, good for text processing, systems and web applications

Python – combines object-oriented and functional programming, used as a scripting language

Ruby – combines object-oriented and functional programming, has similarilites with Perl, Python, Smalltalk, Lisp. Supports meta-programming (called reflection)