CSCE 4430-5450: Programming Languages - Spring 2013

Instructor: Paul Tarau, Associate Professor - see my home page for contact info and office hours.

Teaching Assistant. Fahmida Hamid. See her course page for contact info and office hours.

First exam: TBD

Final exam: TBD – please complete exit surveys

E-mail : *t a r a u@cs.unt.edu*
WWW : http://www.cs.unt.edu/~tarau
Address: Department of Computer Science, University of North Texas, P.O. Box 311366, Denton, Texas 76203, USA
Phone : Tel : +1-940-565-2806, +1-940-565-2767
Fax : +1-940-565-2799

Description and Objectives:

An advanced programming language course, with emphasis on programming paradigms and language processors - and some of their formal models like Predicate Logic and Lambda Calculus and exhibiting actual implementations of key concepts (recursion, inheritance, unification, backtracking, type inference, infinite and lazy data objects, threads, event-driven and concurrent/distributed programming). The course also provides a glimpse at salient features of modern object oriented languages and an overview of language implementation techniques, run-time systems, garbage collection, interpreters, compilers with emphasis on addressing and memory management in efficient procedural languages like C.

Syllabus (L1..Ln) indicate number of the lecture

- Basics
  - Evolution of Programming Languages L1
  - Programming Paradigms: logic, functional, object oriented, imperative L1
  - Language Specification: Syntax and Semantics L1
  - Language Processors: Interpreters and Compilers L2
  - Models of Computation, Computability, Turing Equivalence L2
- Functional Programming Languages (Haskell)
  - Lambda Calculus and Recursion Theory L3
  - Higher Order Functions L4
  - Working with Fold/Unfold, Map, Zip L4

    - Working with Pattern Matching L4
    - Lazy Evaluation, Computing with Infinite Lists L5
    - Polymorphism, Type inference, Type Classes L5
- Logic Programming Languages (Prolog)
    - Unification and Horn Clause Resolution in Prolog L6
    - Non-determinism and Backtracking with application to Problem Solving L6
    - Practical Prolog: IO, File Operations L7
    - Definite Clause Grammars L7
    - Parsing and Generation L7
    - Meta-Interpreters, Universal Machines L8
- Object Oriented Programming (Scala)
    - Quick Introduction to Scala and Eclipse (our main tools) L9
    - Types: Static vs. Dynamic Type Checking L9
    - Classes, Instances, Objects, Case Classes L9
    - Collections, Iterators and IO operations L10
    - Reflection and Serialization L10
- Low Level Imperative Programming (C)
    - basics: assignment, function calls, lexical scoping, memory representations L11
    - implementing dynamic memory management and garbage collection L11
- Implementing high-level programming languages L12
- Concurrent Programming
    - Multi-threaded programming and Actors L13
    - Distributed Programming, Message Passing, Coordination L14
    - Web Service Architectures and Interactivity L14
- Future trends in Programming Language Design L15

Prerequisites: mandatory (Data Structures)

Recommended books and online materials:

- Scala Tutorials
- A. Tucker & R. Noonan: Programming Languages, Principles and Paradigms, McGraw Hill
- Haskell and Categories
- Doets & van Eijck: The Haskell Road to Logic, Math and Programming
- R. Harper's draft PL book at: http://www.cs.cmu.edu/~rwh/plbook/
- Java for Students: Douglas Bell & Mike Parr, Prentice Hall
- The Art of Prolog by Sterling and Shapiro, MIT Press
- Lambda Calculator

Evaluation:

- 2 Individual Exams: 50%
- Team Project and Assignments (groups of 2-3): 50%

Software and tutorials:

- Scala and Akka
- The Eclipse Open Software Development Platform
- A good prolog tutorial
- Haskell compiler GHC and Prelude library sources
- Functional Pearls
- Java interpreter/compiler on silo and CAS network (java, javac), Java from www.javasoft.com
- Java based Prolog compiler
- Programming Languages Popularity/Impact Index
- 99 Haskell Problems based on 99 Prolog Problems Various Haskell Coding Styles
- Programmers Competency Matrix
- An unconventional view about how learning works
- An unconventional view about research on programming languages
- A view on asking good questions in public forums

Outcomes:

1. Understand key concepts of programming languages, with emphasis on programming paradigms and language processors.
2. Have a practical understanding of commonalities and differences between major programming paradigms.
3. Understand the key object oriented, logic and functional programming concepts.
4. Understand the key concepts of event driven and concurrent programming.
5. Understand the use and implementation of modern programming language concepts like recursion, inheritance, reflection, unification, backtracking, type inference, infinite data objects, and threads.
6. Have some familiarity with domain-specific languages with emphasis on Internet programming languages.