Principles of Programming Languages

עקרונות שפות תכנות

Prof. Mira Balaban

- **Course number :** 202.1.2051
- **Mandatory**
- **Credits :** 5
- **Course Site:** http://www.cs.bgu.ac.il/~ppl132
- **Prerequisites:**
  - Data Structures – 202.1.1031
  - Automata and Formal Languages – 202.1.2011

Course Objectives

This course concerns the interplay between problem modeling and programming languages. Problem solving relies on having good models that are written in a language that can support the modeling concepts. There are four main objectives:

- Learning principles of programming languages: The elements of programming languages; abstraction means in programming languages; formal definition of programming languages – concrete syntax, abstract syntax, operational semantics; program correctness – type checking and type inference systems.
- Learning the essence of program execution by evaluators: Interpreters, compilers.
- Understanding programming paradigms: How the above topics are realized in Functional programming, Logic programming and Imperative programming.
- Learning principles of program design: Abstraction, contracts, modular architecture, testing.

The course is a mixture of theory and practice. All theoretical topics involve implemented software, and all course assignments involve a major amount of programming. The course combines two main approaches in teaching principles of programming languages: Using a single language for demonstrating all language aspects vs. introduction of multiple languages as representatives of different aspects. The course uses the Scheme language for teaching the general theory and practice of language design and implementation. Then, the ML language is used for experiencing with static type inference, and Prolog is used for introducing the Logic Programming paradigm.

Course Requirements

- 4-hours weekly lectures
- 2-hours weekly exercise sessions
- 7 theoretical and programming assignments. Can be individual or in pairs
- Midterm
- Final exam

Detailed Syllabus

1. Functional Programming I – The Elements of Programming

1.1 The Elements of Programming

1.2 Types in Scheme

1.3 Program Design Using Contracts

1.4 Procedures and the Processes they Generate

1.5 High Order Procedures

2 Functional Programming II – Syntax, Semantics and Types

2.1 Syntax: Concrete and Abstract

2.2 Operational Semantics: The Substitution Model

2.3 Type Correctness

3 Functional Programming III - Abstraction on Data and on Control

3.1 Compound Data: The Pair and List Types

3.2 Data Abstraction: Abstract Data Types

3.3 Continuation Passing Style (CPS) Programming

4 Evaluators for Functional Programming

4.1 Abstract Syntax Parser (ASP)

4.2 A Meta-Circular Evaluator for the Substitution Model – Applicative-Eval Operational Semantics

4.3 The Environment Based Operational Semantics

4.4 A Meta-Circular Evaluator for the Environment Based Operational Semantics

4.5 A Meta-Circular Compiler for Functional Programming

5 Static Typing in Functional Programming – Programming in ML

5.1 Type Checking and Type Inference

5.2 Basics of ML: Programming with Primitive Types

5.3 Types in ML

5.4 Lazy Lists (Sequences, Streams)

6 Logic Programming - in a Nutshell

6.1 Relational Logic Programming

6.2 Full Logic Programming

6.3 Prolog

6.4 Meta-circular interpreters for Pure Logic Programming

7 Imperative Programming

7.1 Assignment and Local State

7.2 Operational Semantics for Imperative Programming in Scheme

7.3 Costs and Benefits of Introducing assignment

7.4 Object-Oriented Modeling

7.5 Modeling with Mutable Data

7.6 Language Comparison

References

Lecture notes in
http://www.cs.bgu.ac.il/~mira/ppl-book.pdf

Additional reference books:

1. Structure and Interpretation of Computer Programs, by H. Abelson and G.J. Sussman, MIT Press, 2nd edition, 1996.

2. How to Design Programs, by M. Felleisen, R.B. Findler, M. Flatt and S. Krishnamurthi, MIT Press, 2003.
3. Programming Languages: Application and Interpretation, by S. Krishnamurthi, Version 26.4.2007.
4. ML for the Working Programmer, by L. Paulsen, Cambridge university press, 2nd edition, 1996.
5. Programming in Standard ML, by R. Harper, Carnegie Mellon University, 2011.
6. The Art of Prolog, 2nd edition, by L. Sterling and E. Shapiro, MIT Press, 1994.
7. Programming Language Essentials, by H.E. Bal and D. Grune, Addison-wesley, 1994.