CAS 781 — Functional Programming

2003-01-08

Outline

Instructor: Dr. Wolfram **Kahl**
Department of Computing and Software
McMaster University

Room: ITB-245

E-Mail: kahl@cas.mcmaster.ca

Course Pages: http://www.cas.mcmaster.ca/~kahl/FP/2003/

This is where you find further information, especially concerning tutorial organization and software installation. Electronic versions of the assignment sheets will also be kept there.

It is the student's responsibility to be aware of the information in the course Web pages, and to check regularly for announcements.

Literature:

Main Textbook:

Paul Hudak. *The Haskell School of Expression, Learning Functional Programming through Multimedia.* Cambridge University Press, 2000. ISBN 0-521-64408-9.

Haskell Website: http://haskell.org/

Schedule:

Wednesday, 11:30–13:30; Thursday, 9:30–10:30

Grading:

All examinations in this course will be **Closed Book**. That is, no written or printed material nor a calculator may be used during the examinations.

The **final examination** will be scheduled by the Registrar's Office in the usual way. It will be a closed book examination of three hours duration and cover the material of the lectures, tutorials, handouts, and assignments.

In addition, there will be **one midterm examination**; details to be announced. Technically, this midterms is optional; there will be no opportunity to repeat a missed midterm.

For every student, the course grade is calculated as a weighted average of the grade of the final exam and the grades of those midterm examinations that are better than the final, where these midterms count 30% each.

The instructor reserves the right to conduct any deferred exams orally.

Contents

The major topics are most likely to be presented in an interleaved manner that allows applying theory in practice with less delay.

- Foundations of Functional Programming:
  - Term rewriting systems, lambda-calculus, general concepts of reduction systems, evaluation strategies
  - Type systems, parametric polymorphism, ad-hoc polymorphism, second-order polymorphism, Hindley-Milner typing
- Practice of Pure Functional Programming (in Haskell)
  - Values, simple functions, primitive datatypes, lists, list comprehension
  - Function definitions using pattern matching, recursive functions over lists
  - Repetition and iteration patterns, common higher-order functions like fold, map, filter
  - Type classes and instances
  - Datatype definitions, abstract datatypes, the role of class instances in datatype abstraction
  - Computations and State in the purely functional setting: Monads, I/O
  - Advanced datatype abstraction using multiparameter classes, monad transformers, modular interpreters
  - Principles and examples of combinator libraries; parsing combinators, embedded domain-specific languages
- Practice of Impure Functional Programming (in OCaml)
  - Handling of state and I/O in impure languages
  - Advanced module systems: Module types and functors, comparison with Haskell module and class systems, programming-in-the-large
  - Clean embedding of an Object-Oriented class concept in a functional language
- Implementation of Functional Programming Languages
  - Closures
  - Abstract Machines, Graph Reduction
  - Garbage Collection

Official Statements

Academic Dishonesty

> "Students are reminded that they have to read and comply with the Statements on Academic Ethics and the Senate Resolutions on Academic Dishonesty as found in the Senate Policy Statements distributed at registrations and available in the Senate Office."

Discrimination

> "The Faculty of Engineering is concerned with ensuring an environment that is free of all adverse discrimination. If there is a problem that cannot be resolved by discussion among the persons concerned, individuals are reminded that they should contact the Department Chair, the Sexual Harassment Office or the Human Rights Consultant, as soon as possible."