Course Information

Course structure for Spring 2005

The course focuses on programming languages concepts such as binding, evaluation, and types, exhibited on representative languages from the functional and logic programming paradigms. The goal is to establish thorough working knowledge of two programming languages, ML and Prolog, and to practice the craft of programming in different paradigms. The course is fast-paced, and involves *extensive programming assignments*. Initial familiarity with the procedural and functional paradigm is assumed, as described below.

Here is a detailed list of topics students should expect to have mastered by the end of the course:

- Typed Functional Programming (the programming language ML):
  - Purely functional programs in the ML core language.
  - The type system of ML.
  - Type reconstruction.
  - Higher-order functions for list processing.
  - Data types and their operators.
  - Programming in the point-free style.
  - The ML module language.
- Logic Programming (the programming language Prolog):
  - Declarative and logic programming in general.
  - Prolog search, unification, resolution.
  - Impure features of Prolog.
- Programming Language Design and Implementation:
  - Language syntax:
    - Introduction to scanning and parsing techniques.
  - Language semantics:
    - Introduction to semantics specification techniques.
  - Language implementation:
    - Overview of interpretation and compilation.
    - Exercises in small-language interpretation.

Programming

We use two main programming languages, ML and Prolog; we occasionally refer to and make use of various other programming languages.

For ML, we use the Standard ML dialect, and our reference implementation is Standard ML of New Jersey version 110.52.

For Prolog, our reference implementation is SWI-Prolog version 5.4.5.

Assignments and grading

Several (up to 10) short in-class quizzes (10%) , mid-term exam (15%), final exam (covering the entire course) (25%), up to 10 written homework assignments (50%). You are expected to attend class regularly, read the assigned reading before class, and participate in class discussion.

Attendance

Your attendance at lectures is mandatory. To encourage attendance, quizzes will be unannounced and in-class.

Prerequisites in detail

Experience with developing substantial applications in functional and imperative (especially object-oriented) programming languages is required. UNM CS courses CS 257 *Nonimperative Programming* and CS 351 *Design of Large Programs* provide appropriate background in Scheme and Java, respectively. If you have not used Scheme before, you must take CS 257 before enrolling in CS 451.

Lectures

Mondays & Wednesdays, 4:00 - 5:15, in Mechanical Engineering 218.

Instructor

Darko Stefanovic, office FEC 345C, phone 2776561, email darko@cs.unm.edu — office hours Mondays and Wednesdays, 3:00-3:50.

Teaching assistant

Jack M. Pullikottil, office Mechanical Engineering 435, email jackmp@cs.unm.edu — office hours Tuesdays, Thursdays, and Fridays 4:00-5:00.

Mailing list

A mailing list will be used for class discussion amongst the students and for questions to the instructor and the teaching assistant. It may also be used for administrative announcements. See `http://www.cs.unm.edu/cgi-bin/mailman/listinfo/cs451`.

UNM statement of compliance with ADA

Qualified students with disabilities needing appropriate academic adjustments should contact the instructor as soon as possible to ensure their needs are met in a timely manner. Handouts are available in alternative accessible formats upon request.

Textbooks

(The bookstore has ordered the titles marked with *.)

Required reading

* William F. Clocksin and Christopher S. Mellish: *Programming in Prolog. Using the ISO Standard, 5th edition*, Springer Verlag, 2003, ISBN 3540006788.

* Lawrence C. Paulson: *ML for the Working Programmer, 2nd edition*, Cambridge University Press, 1996, ISBN 0-521-56543-X.

Optional reading: books on ML

* Emden R. Gansner and John H. Reppy (eds.): *The Standard ML Basis Library*, Cambridge University Press, 2004, ISBN 0-521-79478-1.

Matthias Felleisen, Daniel P. Friedman: *The Little MLer*, MIT Press, 1997, ISBN 026256114X.

Michael R. Hansen and Hans Rischel: *Programming Using SML*, Addison-Wesley, 1999, ISBN 0-201-39820-6.

Jeffrey D. Ullman: *Elements of ML Programming, ML97 Edition*, Prentice Hall, 2000, ISBN 0-13-790387-1.

Robin Milner, Mads Tofte, Robert Harper, David MacQueen: *The Definition of Standard ML - Revised*, MIT Press, 1997, ISBN 0262631814.

Optional reading: books on Prolog

William F. Clocksin: *Clause and Effect: Prolog Programming for the Working Programmer*, Springer Verlag, 1997, ISBN 3540629718.

Ivan Bratko: *Prolog Programming for Artificial Intelligence, 3rd edition*, Addison-Wesley, 2001, ISBN 0-201-40375-7.

Krzysztof R. Apt: *From Logic Programming to Prolog*, Prentice Hall, 1997, ISBN 0-13-230368-X.

Optional reading: general books

John C. Mitchell: *Concepts in Programming Languages*, Cambridge University Press, 2003, ISBN 0-521-78098-5.

Michael L. Scott: *Programming Language Pragmatics*, Morgan Kaufmann, 2000, ISBN 1558604421.

Richard Bird and Oege de Moor: *Algebra of Programming*, Prentice Hall, 1997, ISBN 0-13-507245-X.

H. P. Barendregt: *The Lambda Calculus: Its Syntax and Semantics, revised edition*, Elsevier North-Holland, 1984, ISBN 0-444-87508-5.

David A. Watt: *Programming Language Concepts and Paradigms*, Prentice Hall, 1990, ISBN 0-13-728874-3.

Michael J. C. Gordon: *Programming Language Theory and its Implementation*, Prentice Hall, 1988, ISBN 0-13-7304170-X.

Anthony J. Field and Peter G. Harrison: *Functional Programming*, Addison-Wesley, 1989, ISBN 0-201-19249-7.

Homework and programming assignment hand-in policy

Homework assignments are due at midnight of the date assigned, and no credit will be given for late homework. However, each student will have 5 spare days that can be used to postpone one or more homework deadlines by up to a total of 5 days over the entire semester, with no questions asked.

Hand-in mechanisms (usually electronic) will be specified with each assignment.

Programming questions on homework assignments

When an assignment asks you to write a program, that means that you must design a program, type it into a computer, compile (depending on the language at hand), and run, and then submit a listing of the program and its output, making sure that your set of test inputs is carefully designed. The textual layout of the program must be logically sound and aesthetically pleasing. The names used must be descriptive. Code comments and additional documentation must accompany non-trivial programs, and must provide informal argumentation that the program satisfies the specification. Occasionally you may be asked to provide a formal correctness proof of a program—include that proof as part of program documentation.

Cooperation and cheating

Feel free to *discuss* homework assignments with classmates and the instructor. However, *do not look at or copy another's solution*. If a problem appears too difficult, or you lack the background to solve it, you are expected to talk to the instructor promptly. Once you have the background necessary to solve a problem, you must provide your own solution. Exchanging homework solutions is cheating and will be reported to the University administration; students involved may not be permitted to continue in the class.

You are responsible for exercising due diligence in protecting your homework files from unauthorized access. In case two or more students present essentially similar homework, all involved students will be reported to the University administration. Occasionally we will reuse problems that have been used in previous courses. Searching on the web for verbatim solutions is not the best application of your time, and you will receive no credit for it.

Each assignment handed in must be accompanied by the following statement: *"I pledge my honor that in the preparation of this assignment I have complied with the University of New Mexico Board of Regents' Policy Manual."*

Of particular relevance is Section 4.8, which reads as follows.

> **4.8 Subject: ACADEMIC DISHONESTY**
> **Adopted: September 12, 1996**
> **Applicability**
> This policy applies to all students at the University with regard to academic activities and professional activities related to academic work.
> **Definition**
> "Academic dishonesty" includes, but is not limited to, dishonesty in quizzes, tests, or assignments; claiming credit for work not done or done by others; hindering the academic work of other students; misrepresenting academic or professional qualifications within or without the University; and nondisclosure or misrepresentation in filling out applications or other University records.
> **Policy**
> Each student is expected to maintain the highest standards of honesty and integrity in academic and professional matters. The University reserves the right to take disciplinary action, up to and including dismissal, against any student who is found guilty of academic dishonesty or who otherwise fails to meet the expected standards. Any student judged to have engaged in academic dishonesty in course work may receive a reduced or failing grade for the work in question and/or for the course.
> **Implementation**
> The President may establish administrative policies and procedures for implementing this policy, which shall be published in the Pathfinder and the Faculty Handbook, together with this policy.

Lecture Plan Overview (subject to change)

- Organizational meeting and background review (1 lecture; 1 homework)
- Typed functional programming and Standard ML (12 lectures; 6 homeworks)
- Mid-term (1 class period)
- Prolog (8 lectures; 3 homeworks)
- Advanced topics—programming language design and implementation (8 lectures)
- Review (1 class period)

- Final exam (during finals week)

This plan is an indication only of the number of lectures intended for each topic, and not of the order in which the topics will be covered. In particular, some advanced topics will be interleaved with the rest of the material.