

CS 510 Semantics and Types – Winter 2008

Instructor: Andrew Tolmach

120-23 FAB (503) 725-5492

email: apt@cs.pdx.edu

Office Hours: MW 1-2pm & by appt.

Course web page: <http://www.cs.pdx.edu/~apt/cs510types>

Description

This course provides an introduction to the mathematics of program meaning (semantics) using the framework of type systems and typed languages. Topics include operational semantics; inductive proof techniques; the lambda-calculus; type safety; basic and advanced types systems including references, exceptions, and subtyping; types for object-oriented languages, and polymorphic types.

Prerequisites

The theoretical material in the course is self-contained, so there are no specific prerequisites, but a reasonable level of mathematical maturity is desirable. For example, you should be comfortable with proofs by induction.

Programming exercises will be in the OCaml language, so previous exposure to a functional language such as Standard ML (to the level of CS558) or Haskell (as in CS557) is quite desirable.

Energetic students may wish to produce machine-checked proofs using the Coq proof assistant. Prior exposure to automated theorem proving is not required.

Readings

We will use the textbook "Types and Programming Languages," by Benjamin C. Pierce, MIT Press, 2002.

Requirements

There will be weekly homework assignments, a midterm, and a final exam (probably closed book). The homework assignments will include both theory problems and short OCaml programming exercises. There will also be optional Coq proving exercises. Since answers are provided to many assignment questions in the back of the book, the assignments will be scored simply based on whether you turn them in or not; hence, they won't count for much of the grade. But doing them will be essential preparation for the exams.

The course grade will be distributed as follows:

Homework	20%
Midterm	35%
Final	45%

Although it will not be formally assessed, class participation is strongly encouraged, and may affect borderline grades.

You are strongly encouraged, though not required, to typeset your homework solutions using `latex`.

Computing Facilities

Some of the homework exercises will require use of the OCAML language; any recent version will do. Ocaml is installed as a package (`ocaml`) on the CS Solaris machines and is available by default on the CS linux systems. It is also very easy to install on your own personal machine (and doesn't require many resources). See the course web page for pointers.

For the Coq exercises, you'll want a recent version (8.1 or better) of the toolset. Coq is available by default on the CS linux systems and should soon be available on the CS Solaris systems. You can also install it on your own personal machine; see the course web page for pointers.

Latex is easily available everywhere.

Individual Work

It is permitted (even encouraged) for you to work together on homework assignments. However, all homework submissions must be written up (or typed in) individually; an important part of the course is learning how to write down theoretical arguments, even after they are clear in your own mind.

Exams must be completed individually without any collaboration. Cheating on an exam will result in an automatic zero grade and the initiation of disciplinary action at the University level.

Disabilities

If you are a student with a disability in need of academic accommodations, you should register with Disability Services for Students and notify the instructor immediately to arrange for support services.

Tentative Schedule

This schedule is highly subject to change. You should always attempt to do the reading *before* the relevant class meeting.

<i>dates</i>	<i>Pierce chapters</i>	<i>topics</i>
Jan 8 & 10	1,(2),3,4	Introduction; Syntax and Operational Semantics
Jan 15 & 17	5,6,7	Untyped lambda-calculus
Jan 22 & 24	8,9,10	Types; Simply-typed lambda-calculus
Jan 29 & 31	11	Extensions
Feb 5 & 7	12,13,14	Normalization; References and Exceptions
Feb 12		Midterm (in-class)
Feb 14	15	Subtyping
Feb 19 & 21	16,17,19	More subtyping; Object-oriented languages
Feb 26 & 28	20,22	Recursive Types; Type Reconstruction
Mar 4 & 6	23,24,25	Universal and Existential Types
Mar 11 & 13	29,30	Higher-order Systems
Mar 20		Final Exam (10:15-12:05)