Special Topics—Modelling semantics with ML

Prerequisites: CPSC 320 and CPSC 370, or CPSC 340 and CPSC 370 with CPSC 320 as a co-requisite, or permission of the instructor

Instructor: Dr. David Casperson

Office: Library 5-444

Telephone: (250)960-6672

Electronic mail address: `casper@unbc.ca`

Lectures: MWF 11:30–12:20 Room 5-152

Text: *Elements of ML Programming: ML97 Edition* by Jefferey D. Ullman.

Reference:

- *First Leaves: A Tutorial Introduction to Maple V* by Bruce W. Char, Keith O. Geddes, *et. al.*
- *Maple V Reference Manual* ***??nth edition*** by Bruce W. Char, Keith O. Geddes, *et. al.*
- *ML-Yacc User's Manual Version 2.3* by David R. Tarditi and Andrew W. Appel.
- *A lexical analyzer generator for Standard ML.* by Andrew W. Appel, James S. Mattson, and David R. Tarditi. `http://cm.bell-labs.com/cm/cs/what/smlnj/doc/ML-Lex/manual.html`
- *Principles of Programming Languages* by R.D. Tennent.
- *others I haven't yet found*

Tentative Grading:				
	Exam 1	:	25%	Wednesday, 9 Feb (tentative)
	Exam 2	:	25%	Monday, 20 Mar (tentative)
	Project	:	15%	
	(Final) Exam 3	:	35%	

Overview: We'll look at functional programming in Standard ML '97; how we can use this to directly model denotational semantics of conventional programming languages; and hopefully use ML to look at the semantics of computer algebra systems.

Syllabus: A review of functions and partial functions from CPSC 141. The notion of Currying.

Basic Standard ML. Builtin types and literals. Lists. Declarations. Function declarations and function values. Pair types and function types.

Recursion, tail recursion, and accumulator arguments. Text and input and output.

Representing partial functions in ML. Options. Exceptions and exception handling.

The idea of denotational semantics. Stores, and environments. Concrete and abstract syntax. Datatypes. Polymorphic datatypes. Multi-clause functions and type deduction. Modelling denotational semantics in ML.

Higher order functions and currying. Continuations and continuation semantics.

LALR(1) grammars. Using ML-Yacc and ML-Lex. Pretty-printers. Lazy evaluation.

Large scale programming in ML. Structures, signatures, and functors. Opaque and transparent signatures. Local blocks and abstract datatypes.

An introduction to the semantics of Maple. Maple datatypes. The Parse-Simplify-Evaluate-Simplify loop. Simplification rules. Last-name evaluation. Interaction between `evalstat`, `eval`, `evaln`, `evalb`, `evalapply`, and the simplifier. Abstract and concrete grammars for Maple. Connections between the language and kernel.