CMPT 442 & 823
Compiler Construction

Christopher Dutchyn

August 26, 2013

This session shares lectures between CMPT442, a senior undergraduate course, and CMPT823, an introductory graduate course. Students taking the course under either number will complete a substantive software engineering project – an actual compiler for a full programming language. As a capstone course, *Compiler Construction* correlates, refines, and solidifies the students' understanding of most aspects of computer science:

- *code generation* solidifies an understanding of hardware
- *system call interfaces* reinforce the abstractions offered by operating systems
- *abstract syntax trees* and *flow-graph*s demand advanced data structures and algorithms
- *symbol tables* and *escape analysis* profit from knowledge representation
- *intelligible error messages* stress the importance and difficulty of human-computer interaction
- *register allocation* gives an everyday example of NP-completeness and computational complexity
- *optimizing code* and *search*ing for pareto-optimal solutions apply fundamental artificial intelligence techniques
- *type checking* pushes the boundaries of theorem-proving and discrete mathematics
- *parsers* of varying power draw on central abstractions from computational theory
- *reliability* demands strong software engineering practices: testing, version control, and configuration management.

The fundamental difference in workload between the two levels is that the undergraduate students will finish with the final examination; whereas the graduate students will avoid the midterm exam, will read and summarize a topical research paper, and extend their course project with additional facility of specific interest.

From the course catalogue:

> CMPT 442.3 2(3L) **Compiler Design and Implementation**
>
> Introduction to the systematic construction of a compiler: context-free and regular grammars, scanners, attribute grammars, parsing, syntax trees, runtime organization, symbol tables, internal representations, compile-time error handling, semantic analysis, storage allocation, code generation, linking, byte code, interpreters.
>
> Students will use compiler construction tools in a term project.
>
> **Formerly:** CMPT 429.
>
> **Prerequisite(s):** CMPT 360. CMPT 340 recommended.
>
> **Note:** Students who have credit for CMPT 429 may not take this course for credit.

and

> CMPT 823.3 1/2(3L) **Compilers**
>
> The definition and classification of formal grammers. A discussion of regular and context-free grammars with their relationships to automata. Precedence, operator precedence, LR(k) and LALR(k) grammars with their associated syntactic analysers, symbol table techniques, intermediate forms of source programs, run-time organization, code generation and optimization. Interpreters and their relation to the compilation process. Introduces translator writing systems and compiler-compilers.

1 Staff

Instructor: Chris Dutchyn

> **email** `mailto:dutchyn@cs.usask.ca`
>
> **web** `http://www.cs.usask.ca/faculty/cjd032`
>
> **office** Thorvaldson 178.2

2 Meetings

We will be meeting for lectures in room THORV 205A, on Tuesdays and Thursdays from 11:30AM to 12:50PM. Important events during the course term include:

- **September 5** – first instructional day.
- **September 18** – last day for registration changes.
- **October 11** – reading day: no classes.
- **October 14** – Thanksgiving day: no classes.
- **November 11** – Remembrance day: no classes.
- **November 15** – last day to withdraw without penalty.
- **December 4** – last instructional day.
- **December 6–21** – final exam period.

The CMPT 442 final examination date will be scheduled by central timetabling, but will occur during the December 6 to December 21 interval.

Instructor office hours are

- Wednesday, and Thursday from 10:00–11:00AM in the instructor's office. Please knock if the door is closed.

3 Other Communication

The course is supported by the Moodle instructional system, located at

`http://moodle.cs.usask.ca`

There will be

- course content (notes) of various forms,
- assignments, including the assignment submission form,
- a bulletin board discussion facility,

The material will be released as the course progresses. Please familiarize yourself with the layout immediately. Projects will be hosted on the department github

`http://git.cs.usask.ca`

with separate teams given separate projects. One of the early decisions will be team selection.

4 Grading

Intangibles may count in the determination of your grade. Regular attendance and productive classroom participation may slightly ameliorate some weaknesses elsewhere; the converse is also true.

Furthermore, the following University policies must be adhered to:

1. All students must be properly registered in order to attend lectures and receive credit for this course.
2. All students must accept and adhere to the Computer Science departmental academic honesty policy.

This course is a split class, including CMPT 442 (undergraduate) and CMPT 823 (graduate) students. The work expected of each set of students, and their grading criteria will be different.

4.1 Undergraduate

The work is divided between examinations and a major project in four stages (with 10%, 15%, 15%, and 15%) where the students will write a compiler for an expressive, imperative programming language. This work will be done in groups of three (or less).

An in-class midterm, (15%), and a final examination (30%) complete the course requirements.

	CMPT 442	
Item	Description	Weighting
Project Stages	(four: 10%, 15%, 15%, 15%)	55%
Mid-term Examination	(50 minutes, in-class)	15%
Final Examination	(180 minutes)	30%
Total		100%

4.2 Graduate

The majority of the coursework is completing a project in pairs. The first four stages are identical to those of the undergraduate students with the same weight: 10%, 15%, 15%, and 15%. Graduate students will be expected to select one optional topic which they will implement in their compiler (15%). These selections will determine the optional topics given at the end of the term.

Starting in October, the graduate students will be offered a selection of research papers on topics related to the course, and expected to read and summarize one these.

CMPT 823		
Item	Description	Weighting
Project Stages	(five: 10%, 15%, 15%, 15%, 15%)	70%
Paper Review		10%
Final Exam		20%
Total		100%

5 Topics

Part I of the text book, and selected topics from part II as time permits.

Table 1: Regular Topics

Chapter	Description	Lectures	Assignment
§1	INTRODUCTION, TEST-FIRST DESIGN	*1*	
	INTERPRETERS, PARTIAL EVALUATION	*1*	
§2	LEXICAL ANALYSIS	*2*	
§3	PARSING	*2*	
§4	ABSTRACT SYNTAX	*1*	**milestone 1**
§5	SEMANTIC ANALYSIS	*3*	
§6	ACTIVATION RECORDS	*3*	**milestone 2**
§7	TRANSLATION	*2*	
	in-class mid-term examination	*1*	
§8 & §17	BLOCKS, TRACES, FLOW GRAPHS	*4*	
§9	RISC MACHINES, INSTRUCTION SELECTION	*2*	**milestone 3**
§10	LIVENESS	*1*	
§11	REGISTER ALLOCATION	*1*	
§12	PUTTING IT TOGETHER	*2*	**milestone 4**
	Total	*26*	

6 Textbooks

6.1 Required

Appel — *Modern Compiler Implementation in ML*, Cambridge University Press, 1998 (reprinted 2004).

This text the primary material for the course; there will be assigned readings and exercises from it. It also contains the description of our language, Tiger, and the various modules supplied with the project.

Table 2: Optional Topics

Chapter	Description
§13	GARBAGE COLLECTION
§14	OBJECT ORIENTED LANGUAGES
§15–§16	FUNCTIONAL LANGUAGES and POLYMORPHIC TYPES
§17–§18	DATAFLOW ANALYSIS and LOOP OPTIMIZATION
§X	PARALLEL PROGRAMMING LANGUAGES
§Y	ASPECT ORIENTED LANGUAGES

6.2 Software

`http://www.smlnj.org` — the current Standard ML of New Jersey implementation (v. 110.76 as of this time).

`http://www.cs.princeton.edu/~appel/modern/ml` — the support files for the course textbook.

`http://www.pllab.riec.tohoku.ac.jp/smlsharp/?SMLUnit` — a version of SMLUnit.

6.3 Supporting

Aho, Lam, Sethi, and Ullman — *Compilers*, 2ed., Addison-Wesley, 2007 (aptly called "*the dragon book*").

Felleisen and Friedman — *the Little MLer*, MIT Press, 1998.

Fisher, LeBlanc, and Cytron — *Crafting a Compiler*, 2ed., Addison Wesley Longman, 2009.

Muchnick — *Advanced Compiler Design and Implementation*, Morgan Kaufman, 1997.

Paulson — *ML for the Working Programmer*, 2ed., Cambridge University Press, 1996.

Tremblay and Sorenson — *the Theory and Practice of Compiler Writing*, McGraw Hill, 1985.

- *Aug. 26, 2013: initial 2013–2014 draft*