CISC 465/3.0 Foundations of Programming Languages[1]

Organization

CISC 465/3.0 will be offered as a reading and tutorial course this year. Students will be assigned to meet with the instructor in small (2-3 person) groups once a week. At each session, students will be given a reading assignment, typically some sections in the text, and exercises to do. At the following session, the material and solutions will be reviewed. Between sessions, students may consult the instructor for assistance. There will not be any tests or exam: weekly letter grades will be assigned by the instructor and used to determine the final course grade.

Prerequisites

The following are essential prerequisites:

- knowledge of elementary discrete mathematics, especially (i) elementary logic; (ii) sets, functions and relations; and (iii) partially-ordered sets.
- experience with high-level programming languages, including both a procedural language such as C, Turing, or Python, and a functional language such as LISP or Haskell.
- exposure to a formal system for developing or verifying imperative programs, such as Hoare's logic or the refinement calculus.

CISC courses 121, 203, 204, 260 and 223 (or equivalents) would satisfy these requirements.

Students who don't feel confident of their background in discrete mathematics (sets, functions, relations, partial orders) and elementary logic should do some review as soon as possible in any suitable book, such as

S. Lipschutz and M. Lipson. *Schaum's Outline of Discrete Mathematics* McGraw-Hill, third edition, 2009.

K. Devlin. *Sets, Functions, and Logic: An Introduction to Abstract Mathematics*. Chapman and Hall, second edition, 1992. Call number: QA37.2.D48.

D. Velleman. *How to Prove it: a Structured Approach*. Cambridge University Press, 1994. Call number: QA9.V38.

Instructor

The instructor is Dr. Bob Tennent, room 541, Goodwin Hall; his e-mail address is `rdt@cs.queensu.ca`, and his telephone number is 613-533-6060; FAX: 613-533-6513.

[1] `http://www.cs.queensu.ca/home/cisc465`

Motivations

Designers, implementers, and serious users of a programming language need a complete and accurate understanding of the *semantics* (the intended meaning) as well as the *syntax* (the form) of every construct of that language. There is a well-developed and widely-known mathematical theory of formal languages which supports accurate description of (the context-free aspects of) the syntax of programming languages, and correct implementation of scanners and parsers. But the descriptions of context-dependent aspects of syntax (such as scope and type checking) and of semantics in reference manuals and language standards are almost always inadequate because they are based primarily on implementation techniques and intuition.

Rigorous mathematical theories of the semantics and context-dependent aspects of the syntax of programming languages are needed to support correct description and implementation of languages, systematic development and verification of programs, analysis of existing programming languages, and design of new languages that are simpler and more regular. This course is an introduction to these theories.

Text

The main textbook will be (the first six chapters of)

R. D. Tennent. *Semantics of Programming Languages*. International Series in Computer Science. Prentice-Hall International, 1991. Available from 2002 as a Print-on-Demand Edition from Pearson Education. Call number: QA76.7.T473.

The most recent printing of this book has a blue and teal cover; however, earlier printings with red and white covers have identical content.

The lectures will follow the text fairly closely, but not all of the material in the six chapters will be covered, and the order of presentation may be slightly different.

Additional References

The text was written for graduate students and is relatively terse. The following are recommended for supplementary reading.

J. C. Reynolds. *Theories of Programming Languages*. Cambridge University Press, 1998. Call number: QA76.7.R495.

G. Winskel. *The Formal Semantics of Programming Languages: An Introduction*. The MIT Press, Cambridge, Mass., and London, England, 1993. Call number: QA76.7.W555.

H. R. Nielson and F. Nielson. *Semantics with Applications, a Formal Introduction*. John Wiley, Chichester, England, 1992. Available on-line[2]. Call number: QA76.7.N54.

D. A. Schmidt. *Denotational Semantics, A Methodology for Language Development*. Allyn and Bacon, Newton, Massachusetts, 1986. Available on-line[3]. Call number: QA76.7.S34

All of these books are available at Douglas (Science and Engineering) library. The text is on reserve for 3-day loans. The Nielson and Schmidt books are downloadable; follow the links.

[2] `http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.html`

[3] `http://www.cis.ksu.edu/~schmidt/text/densem.html`

Course Outline

Introduction

- context-free and context-dependent syntax
- denotational, operational and logical approaches to semantics

Simple Imperative Languages

- denotational semantics of
  - elementary control structures
  - variables and assignment
  - while loops
- operational semantics of a simple imperative language
- program specifications and program proofs
- validity of program-correctness axioms and rules

Simple Functional Languages

- denotational semantics of
  - function application and definition
  - recursion
- domain theory
- operational semantics of a simple functional language
- quantification and substitution
- validity of program equivalences and derivation rules

Procedural Languages

- combining the syntax and semantics of imperative and functional languages

Learning Outcomes

Students will be able to specify the syntax and semantics of programming-language features in imperative and functional languages, including context-dependent syntax (scope and type-checking) and recursive definitions.

Students will be able to use semantic definitions to prove semantic equivalences and soundness and completeness of operational semantics, and validate program-correctness axioms and rules, using, where appropriate, inductive techniques.

Learning outcomes are assessed by assigning weekly exercises.

Academic Integrity

Students are responsible for familiarizing themselves with the regulations concerning academic integrity and for ensuring that their assignments conform to the principles of academic integrity. Information on academic integrity is available in the Arts and Science Calendar[4] and on the Arts and Science website[5].

In CISC 465, students are encouraged to collaborate in learning the material and understanding the assignment questions. Joint solutions may be submitted, provided all the participants are named and have contributed roughly equally to the submitted solution.

Departures from academic integrity include plagiarism, use of unauthorized materials, facilitation, forgery and falsification. Actions which contravene the regulation on academic integrity carry sanctions that can range from a warning or the loss of grades on an assignment to the failure of a course to a requirement to withdraw from the university.

Disability Accommodation

Queen's University is committed to achieving full accessibility for persons with disabilities. Part of this commitment includes arranging academic accommodations for students with disabilities to ensure they have an equitable opportunity to participate in all of their academic activities. If you are a student with a disability and think you may need accommodations, you are strongly encouraged to contact the Disability Services Office (DSO) and register as early as possible. For more information, including important deadlines, please visit the DSO website[6].

[4] `http://www.queensu.ca/artsci/academic-calendars/regulations/academic-regulations/regulation-1`

[5] `http://www.queensu.ca/artsci/academics/undergraduate/academic-integrity`

[6] `http://www.queensu.ca/hcds/ds/`