General Information

Instructor: Matthew Fluet
E-mail: mtf@cs.rit.edu
Office hours: Mo 10:00am – 11:00am; GOL-3555
              Tu 10:00am – 12:00pm; GOL-3555
              We 10:00am – 11:00am; GOL-3555
              or by appointment

Lectures: Section 01 MoWeFr 1:00pm – 1:50pm; GOL-1435

Website: http://www.cs.rit.edu/~mtf/teaching/20131/plt
          http://mycourses.rit.edu

Course Description

This course is an introduction to the formal study of programming languages, demonstrating important intellectual tools for the precise description of programming languages and investigating the essential features of programming languages using these tools. Topics include: dynamic semantics (such as operational semantics); static semantics (such as type systems); proofs by induction on structures and derivations; formal treatment of essential programming-language features (such as assignment, scope, functions, objects, and threads). Both written and programming assignments will be required.

Prerequisites

- ((CSCI-603 Advanced C++ and Program Design with B or better and CSCI-605 Advanced Java Programming with B or better) or (CSCI-141 Computer Science I and CSCI-142 Computer Science II and CSCI-243 The Mechanics of Programming)) and CSCI-661 Foundations of Computer Science Theory
- or permission of instructor

Course Goals

The precise description of the semantics of programming languages is required to thoroughly understand the meaning of computer programs.
This course studies the formal semantics of programming languages. Students will learn about important intellectual tools such as operational semantics and type systems and will investigate essential features of programming languages using these tools. While the focus is on formal models of small languages, the applicability of these formal models to “real” programming languages will be demonstrated. Students will gain an appreciation of the design decisions (and design mistakes) in extant programming languages and will be prepared to study the programming-languages research literature.

This course does not cover tools and techniques for describing the concrete syntax of programming languages (e.g., scanners and parsers); such topics are covered in CSCI-741 Compiler Construction.

Well-prepared undergraduate students may enroll in this course with the instructor’s approval.

Topics

- Abstract syntax
- Judgements and inference rules
- Operational semantics
- Lambda calculus
- Simply-typed lambda calculus
- Type-safety proof (preservation, progress, substitution)
- Simply-typed lambda calculus extensions
- Subtyping
- Parametric polymorphism, recursive types, and existential types
- Concurrency with shared-memory and with message-passing
- Advanced concepts in object-oriented programming

Note: The order in which topics are discussed in lectures will likely differ from that given above. Furthermore, not all topics will receive equal (or, possibly, any) time. Exams will only cover topics explicitly discussed in lecture or in an assigned reading.

Course and Program Outcomes

Course learning outcomes:

- Students will be able to give precise definitions of programming-language features using operational semantics and type systems.
  Program outcome(s): 2
  Evaluation: exams, homework assignments

- Students will be able to translate programming-language specifications from mathematical notation to code.
  Program outcome(s): 2
  Evaluation: exams, homework assignments
• Students will be able to prove properties of inductively defined sets (e.g., well-typed programs).
  Program outcome(s): 2
  Evaluation: exams, homework assignments

• Students will be able to effectively make use of the research literature in programming languages.
  Program outcome(s): 2, 4
  Evaluation: exams, homework assignments

Program Outcomes:

• (CS Graduate Program Outcome 2) Demonstrate a depth of knowledge in a selected area in the discipline.
• (CS Graduate Program Outcome 4) Pursue professional positions or further graduate studies.

Grades

Grades will be assigned based on the following grading scheme:

  Attendance & Participation: 5%
  Homework Assignments: 50%
  Technical Perspective Assignment: 10%
  Mid-term Exam 1: 10%
  Mid-term Exam 2: 10%
  Final Exam: 15%

Important Dates

  September 27 (Fri.): Mid-term Exam 1 (in class)
  November 11 (Fri.): Mid-term Exam 2 (in class)
  December 18 (Wed.): Final Exam (12:30pm – 2:30pm; GOL-1435)
Text Books

Suggested:

Title: Types and Programming Languages
Author: Benjamin C. Pierce
Publisher: The MIT Press
ISBN: 978-0262162098
Website: [http://www.cis.upenn.edu/~bcpierce/tapl/](http://www.cis.upenn.edu/~bcpierce/tapl/)
RIT Library e-book: [http://albert.rit.edu/record=b1889507~S3](http://albert.rit.edu/record=b1889507~S3)

Title: Practical Foundations for Programming Languages
Author: Robert Harper
Publisher: Cambridge University Press
ISBN: 978-1107029576

Additional:

Title: Advanced Topics in Types and Programming Languages
Editor: Benjamin C. Pierce
Publisher: The MIT Press
ISBN: 978-262162289
Website: [http://www.cis.upenn.edu/~bcpierce/attapl/](http://www.cis.upenn.edu/~bcpierce/attapl/)
RIT Library e-book: [http://albert.rit.edu/record=b1937123~S3](http://albert.rit.edu/record=b1937123~S3)

Title: The Formal Semantics of Programming Languages
Author: Glynn Winskel
Publisher: The MIT Press
ISBN: 978-0262731034

Title: Transitions and Trees
Author: Hans Hüttel
Publisher: Cambridge University Press
ISBN: 978-0521147095

Course Policies

Attendance & Participation

Students are required to attend and expected to participate in class. Participation means being an engaged student: asking and answering questions, not simply attending class.

The use of cell phones and audio players is prohibited during class. If you must take a phone call, please leave the classroom immediately and do not return until you have ended the phone call.

The use of a laptop (or notebook or netbook) computer is permitted during class only for the purpose of taking notes. Persistent use of a laptop for other activities will result in 0 credit for your Attendance & Participation grade.
Assigned readings should be completed before the lecture section. You are responsible for the material in assigned readings, whether covered during lecture or not.

**Mid-term Exam**

There will be two mid-term exams; see above for the date.

The mid-term exams must be taken at their scheduled times. Make-up mid-term exams will not be administered, unless exceptional circumstances have been discussed with the instructor in advance of the exam date and/or other arrangements have been made.

**Final Exam**

There will be a final exam; see above for the date. The final will be comprehensive and will cover material from the entire course, including readings, lectures, and assignments.

The final exam must be taken at its scheduled time. Any exam conflicts must be reported to the instructor by the end of Week 10 (see the [RIT Final Examination Policies](#)).

**Late Policy**

Assignments will generally be due at the beginning of a class period (whether submitted electronically or in person).

Assignments are to be submitted on time. However, to accommodate the occasional difficulty with meeting an assignment due date, each student begins the term with five “extension tokens.” By spending an extension token, you will receive a 24-hour extension on a single assignment. To spend an extension token, you must e-mail the instructor before the assignment is due; you cannot spend an extension token after an assignment’s due date has passed. You may spend at most one extension token on a single assignment. After spending five extension tokens, late assignments will not be accepted.

**Regrading**

After a graded exam or assignment has been returned, you have **one week** to bring any questions about grading to the instructor’s attention. No grade adjustments will be made after this time.

**Academic Integrity**

As with all courses, the [RIT Honor Code](#) and [RIT Academic Honesty Policy](#) apply. See the Department of Computer Science’s statement on [academic integrity](#) for more details.
In this course, all submitted work must be your own work (i.e., written or programmed by you alone, unless explicitly stated otherwise) and must include acknowledgments of any collaborators or sources (other than course text books or handouts) used to produce your submission.

You are encouraged to discuss course material with other students. Discussion of assignments is also allowed, but sharing solutions or code is not allowed.

**Common Course Policies**

See the Department of Computer Science’s [Common Course Policies](#) for more details about rescheduling an exam, course withdrawal, disability services, and academic integrity.

**Technical Perspective**

Each student will write a “Technical Perspective” for a recent programming-languages research paper. A “Technical Perspective” is a two page paper that explains the importance of the research and the specific contributions of the paper; it is written by an expert for non-expert computer scientists and helps to place the paper within a larger research context.

You will need to choose a research paper (a list of suitable papers will be provided), read and understand the paper (which, in turn, will require reading additional background material), and write a “Technical Perspective.” Be aware that writing a short “Technical Perspective” is more challenging than writing a long paper summary.

For more information, see [tech_perspec.pdf](#) / [tech_perspec.html](#).

**Disclaimer**

I reserve the right to make any changes to the syllabus as I deem necessary throughout the course. Minor changes, such as assignment due dates, will be announced orally during class and posted on the course mailing list and home page. Major changes, such as grading percentages, will additionally be provided in writing.

**Acknowledgements**

Portions of this course material based upon similar courses offered at University of Washington (Dan Grossman), Cornell University (Andrew Myers, Dexter Kozen, Nate Foster), Indiana University (Amal Ahmed), Harvard University (Greg Morrisett).