Syllabus

Tiago Cogumbleiro

CS 720: Logical Foundations of Computer Science

Fall 2018

Course information

• Room: M-1-0208
• Schedule: Tuesdays & Thursdays 12:30noon to 1:45pm
• Office hours: Tuesdays & Thursdays 2:30pm to 4:00pm

Instructor contact

• Email: Tiago.Cogumbleiro@umb.edu
• Office: S-3-088
• Phone #: 617.287.6479

Reading material

• Software Foundations. Volume 1 and 2. Benjamin Pierce et al. 2018. softwarefoundations.cis.upenn.edu

Course description

What makes a functional language functional? What makes an imperative language imperative? Can we prove that all program executions are absent of undefined behavior? Can we prove that a compiler preserves the intent of the source program?
This course will introduce the formalization of the semantics of a programming language. We will use a rigorous mathematical formalism to describe a computation model (eg, a programming language) and its properties (eg, avoiding undefined behavior).

The objective of this course is to specify a functional and imperative language with pre- and post-conditions. We will use the Coq Proof Assistant to specify the language, and prove the properties of the programming language we are specifying.

One of the underlying goals of the class is to appeal to practitioners (software engineers) as well as researcher. To this end, we will learn a lightweight testing technique, called property testing, that leverages properties of a computation model (eg, a data structure or a component) to improve the quality of our software.

Topics covered

- basic concepts of logic
- computer-assisted theorem proving (Coq)
- functional programming
- logic programming
- operational semantics
- static type systems
- Hoare logic

Prerequisites

The course is taught “from the ground up,” so it does not assume a specific background in logic or programming. However, familiarity with functional programming and rigorous mathematical proofs is advantageous.

If you are unsure about meeting the requirements for this course, I recommend doing the Coq exercises in the Basics.v chapter of the Software Foundations book, Volume 1.

Course work and grades

- Homework: 75%
- Presentation: 15%
- Participation: 10%
Software requirements

Students are expected to have access to CoqIDE 8.7.2.

Homework assignments consist of Coq programs that will be submitted to the instructor unless stated otherwise.

Attendance

Attendance is required. All students are responsible for knowing everything that is covered during class meetings, including announcements. If you must be absent from a class meeting, make arrangements with another student to find out what you missed.

Homework

No late homework will be accepted.

Any homework may be revised and resubmitted once, up to one week after the original deadline. (If you didn’t submit on time, your original score will be 0.) The grade for that homework will be the average of the two submissions.

All homework will be weighted equally. Your lowest homework score will be dropped.

You may not collaborate with anyone else on any homework. Each homework represents your own, individual work.

It is acceptable to discuss the concept in general terms, but unacceptable to discuss specific solutions to any homework assignment.

On auto grading

Grading of Coq exercises is done automatically. Thus, in order for a homework to be graded, it needs to be accepted by Coq. Use Admitted to allow incomplete proofs and definitions.

Presentations

Every student must present one of two:

- One chapter of the textbook to the class (60 minutes). The instructor will still publish the slides of that chapter.
• One paper on the subject of formalizing the semantics of a programming language or system (20 minutes presentation). The student may suggest a paper (or request one suggestion).

Participation

The participation component in the evaluation corresponds to student participation in classes and in the online forum. Participation does not just mean just answering questions correctly. Discussion and questions, either posed online or in class, are encouraged and counted toward participation.

Each reasonable student intervention (in the class or online) yields 1 point. If the student reaches 14 points, they are graded the full mark of participation.

Accommodations

This class seeks ways to become a working and evolving model of inclusion and universal design for all participants. Individuals with disabilities of any kind (including learning disabilities, ADHD, depression, health conditions), who require instructional, curricular, or test accommodations are responsible for making such needs known to the instructor as early as possible. Every effort will be made to accommodate students in a timely and confidential manner. Individuals who request accommodations must be registered with the Ross Center for Disability Services, which authorizes accommodations for students with disabilities. If applicable, students may obtain adaptation recommendations from the Ross Center for Disability Services, M-1-401, (617-287-7430), [www.rosscenter.umb.edu](http://www.rosscenter.umb.edu). The student must present these recommendations and discuss them with each professor within a reasonable period, preferably by the end of Drop/Add period.

Student Conduct

Students are required to adhere to the University Policy on Academic Standards and Cheating, to the University Statement on Plagiarism and the Documentation of Written Work, and to the Code of Student Conduct as delineated in the catalog of Undergraduate Programs, pp. 44-45, and 48-52. The Code is available online at: [www.umb.edu/life_on_campus/policies/community/code](http://www.umb.edu/life_on_campus/policies/community/code)

Schedule

Below is the tentative schedule of the course.
Week 1:
- Lecture 1, Sept 4 (Tue): Introduction (Basics.v)
- Lecture 2, Sept 6 (Thu): Induction (Induction.v)

Week 2:
- Lecture 3, Sept 11 (Tue): Data structures (Lists.v)
- Sept 12 (Wed): Homework 1 due: Basics.v
- Lecture 4, Sept 13 (Thu): Lists, Polymorphism (Poly.v)

Week 3:
- Lecture 5, Sept 18 (Tue): Tactics (Tactics.v). Homework 2 due: Induction.v, List.v
- Lecture 6, Sept 20 (Thu): More tactics (Tactics.v).

Week 4:
- Lecture 7, Sept 25 (Tue): Logic in Coq (Logic.v). Homework 3 due: Poly.v
- Lecture 8, Sept 27 (Thu): Logic in Coq (Logic.v). Homework 4 due: Tactics.v

Week 5:
- Lecture 9, Oct 2 (Tue): Inductively defined propositions (IndProp.v).
- Lecture 10, Oct 4 (Thu): Inductively defined propositions and proof objects (IndProp.v, ProofObjects.v) Homework 5 due: Logic.v

Week 6:
- Lecture 11, Oct 9 (Tue): Total and partial maps, modeling an imperative programming language (Maps.v, Imp.v, ImpParser.v, ImpCEvalFun.v).

Week 7:
- Lecture 13, Oct 16 (Tue): Program equivalence (Equiv.v).
- Lecture 14, Oct 18 (Thu): Hoare Logic (Hoare.v). Homework 7 due: Imp.v

Week 8:
- Lecture 15, Oct 23 (Tue): Hoare Logic (continued). (Hoare2.v)
- Lecture 16, Oct 25 (Thu): Small-step operational semantics (Smallstep.v) Homework 8: Equiv.v

Week 9:
- Lecture 17, Oct 30 (Tue): Small-step operational semantics (Smallstep.v).
- Lecture 18, Nov 1 (Thu): Types (Types.v). Homework 9: Hoare.v, Hoare2.v
Week 10:
- Lecture 19, Nov 6 (Tue): Simply typed lambda calculus (STLC) (Stlc.v).
- Lecture 20, Nov 8 (Thu): Properties of the STLC (StlcProp.v). **Homework 10:** SmallStep.v

Week 11:
- Lecture 21, Nov 13 (Tue): Extending the STLC (MoreStlc.v).
- Lecture 22, Nov 15 (Thu): Typechecking, Subtyping (Typechecking.v). **Homework 11:** Types.v

Week 12:
- Lecture 23, Nov 20 (Tue): Subtyping (Sub.v). **Homework 12:** Stlc.v, StlcProp.v
- Lecture 24, Nov 22 (Thu): (No class. Thanksgiving recess.)

Week 13:
- Lecture 25, Nov 27 (Tue): Presentations (TBA).
- Lecture 26, Nov 29 (Thu): Presentations (TBA). **Homework 13:** MoreStlc.v

Week 14:
- Lecture 27, Dec 4 (Tue): Presentations (TBA).
- Lecture 28, Dec 6 (Thu): Presentations (TBA). **Homework 14:** Typechecking.v

Week 15:
- Lecture 29, Dec 11 (Tue): Presentations (TBA).
- Lecture 30, Dec 13 (Thu): Course review. **Homework 15:** Sub.v

**List of changes**

- September 18: updated the calendar to reflect the homework due date to be (usually) Thursdays rather than usually on Tuesdays.