COS 598A: Principles of Programming Languages: Spring 2023

David Walker

Princeton University

Formal Description

This course will offer an introduction to principles of programming languages through the definition of little languages such as Kleene algebra, the lambda calculus, and the pi calculus. It will cover topics in denotational and operational semantics of programming languages, techniques for reasoning about programming languages, type systems, logics, and programs, principles of programming language design such as compositionality, orthogonality, soundness, and completeness, proof techniques such as logical relations, and the relationship between programming languages and formal logic.

Schedule

Tuesday, Thursday 1:30-2:50
Room: Friend Center, 111

A detailed tentative schedule is here. It is under constant revision and may change at any time.

Slack

Slack will be used for course communication. The slack workspace is "Principles of Programming Languages Spring 2023." Email the course instructor (dpw@princeton.edu) with "598A" in the title for the slack workspace handle.

Participation

You are expected to participate in class meetings. Participation includes, but is not necessarily limited to, reading papers on time, completing exercises on time, asking useful questions, and participating in class discussion.

Exercises/Assignments

There will be 2-3 assignments. Do the assignments. Print them out. Bring them to class the day they are due. They will be graded in class and then handed in to be reviewed by the prof.

Penalty for late assignment is 30% (don't be late -- instead, do the assignment early). Give an assignment to a friend to turn in in class if you can't make the class. Consideration will be given if there is a significant student illness or other unavoidable problem that renders on-time hand in the impossible.

You may discuss assignments with friends. Try not to give answers away directly but do try to help each other. Please write down the names of anyone you collaborated with or got help from at the top of your assignment.

Projects

All students must do a course project in the 2nd half of the semester. Students may team up and do projects in pairs (ideal) or triples (may be more challenging -- speak with your prof about teams.

The goal of a project is to study some "small" programming language (a "calculus") or programming language feature and its meta-theory. For PhD students taking the class for credit, this study should not overlap significantly with the students work on their general exam or in their central PhD research. Example topics for study include:

- Incremental computation (eg: work by Umut Acar; Adapton)
- Dynamic software update (eg: work by Hicks and others on Kitsune)
- Distributed or concurrent programming and proofs (eg: CRDTs - conflict-free replicated data types, pi calculus, session types, monotonic distributed systems)
- Monotonicity
- Authentication (eg: work by Abadi et al)
- Sequent calculus, logic programming
- Data processing (eg: XML processing such as CDuce, XDuce, XQuery)
- Dependent type systems (eg: LF, calculus of inductive constructions, agda)
- Probablistic programming or program differentiation (eg: probabilistic lambda calculus)
- Computational interpretations of modal logic not studied in class
- Classical logics and computational interpretations
- Substructural logics (eg: linear logic, relevance logic)
- Type inference techniques
- Type and effect systems
- Algebraic effects

Ask your prof for suggested papers.

All projects must contain a written component. All projects will include a project proposal and a final written paper. 5-15 pages is a reasonable length, but content, analysis, thought and creativity are more important than length. The length of the write up may depend upon what other things were done.

Implementation projects may include a shorter write up, but some write up is expected. Implementation projects will be graded in part via a demo to show what has been done --- make arrangements with the professor to give a demo. (See due dates below).

Project topics may involve examining any logic or computational model and its meta-theory

However, projects can take several forms and they may overlap with the lecture you give:

- An implementation project oriented around re-implementing the results of someone else's research paper. The goal of this kind of project is not necessarily to implement something new. It may be to implement a system closely related to some existing system. However, a fresh implementation can often be an excellent learning experience and may lead to new insights, though it does not have to.

- A literature review: Take 2-5 papers not covered in class (and not on your generals list). More papers if they are shorter/less complex. Fewer papers if they are longer/more complex. Explain them. Give new concrete examples. Analyze them. Prove theorems yourself. Draw comparisons. Suggest future work.

- A tutorial that explains some concept related to other students (or faculty!)
    - For theoretical projects, a tutorial may consist entirely of a written document
    - Good tutorials will often synthesize ideas that are scattered across multiple technical papers, extracting and clarifying the most important ideas.
    - Good tutorials include concrete examples and exercises that test a student's knowledge of the topic.
    - All tutorials will include a section that explains the historical evolution of the relevant ideas, citing the appropriate primary sources.

- A standard research project oriented around programming languages and/or formal methods.
    - Standard research projects will include a written description of the work done. This description may take the form of a short workshop paper (6 pages). It may also take other forms, and may include a demo. It may be primarily implementation-oriented. There must be some written component that describes what has been done.

- Some other project. Please discuss with the professor if you have other ideas. Please do not hesitate to suggest a project that engages the course material in some creative new way.

Project Proposal

All students will turn in the project proposal right after the March break. The proposal should be 1-2 pages describing the expected area of study and demonstrating that a student has done some initial work (eg: gathered suitable papers and read intros, looked into programming libraries, etc)

Class Lecture

Each student will lead a lecture in the 2nd half of the semester on some topic in programming languages and/or logic. It would make sense for the lecture to be connected to the topic of your project. Depending on the number of students in the class, students may pair up with each other to create the lecture. Lectures may be based on an interesting paper (or two related papers) or on tutorial materials created by others. Students may assign advanced reading and/or.

At least one week prior to the assigned lecture time, students will arrange a 30 min meeting with the professor to discuss a detailed lecture plan, which should have been prepared in advance.

Lectures will be graded primarily on their educational value to other students in the class. Hence, your goal is to explain a new idea clearly to your peers. The concept should be neither too easy nor too difficult, but "just right" for the class. It is a challenging task!

Grading

Assignments: 20% (30% if there are 3 assignments)
Student Lecture: 20% (5% for proposal)
Project: 50% (10% for proposal)
Participation: 10%

Deadlines

- See the class schedule

Office Hours

After class or 12pm on Fridays in office. On request at different times.