Static Analysis for Software Engineering (version 1)
ECE750-T5 (LEC 043), Winter 2021

Patrick Lam

"Interesting" times. First, a note about the current situation, as of mid-November 2020. I've been on sabbatical for calendar 2020 so I am new to remote teaching. However, I've been in contact with colleagues and students, and I understand that this time has been challenging for everyone. Together, let's try to make the best of this situation. Even in this form, this course remains an opportunity to develop our technical and communication skills. I'm here to support each of you.

I anticipate being physically in New Zealand for this term, due to the pandemic.

Class meetings. Following discussion with the class, I have chosen Tuesdays 21:00-22:30 Eastern Time for class meetings, at least for the duration of Eastern Standard Time (until March 14). We'll check in again near the time change. We will meet in an MS Teams meeting that I have sent out.

Every week, we will discuss 2 or 3 papers, which you've signed up to present. I would like your presentations to be available 24 hours before the class meeting; we should then all view the presentations before the meeting. In our time together, we'll talk about each of the papers, including positives and negatives, and then we'll compare and contrast the papers.

I am also available for individual and group meetings. Just let me know and we'll work out a time that works for all participants.

Setting. Code review is a key technique for ensuring software quality. However, human reviewers have limitations: limited time, limited attention spans, and a limited understanding of the implications of a software modification (due to interactions between parts of a potentially vast codebase).

Computers can successfully carry out many tasks at which humans fail. A goal of my research is to find classes of properties which are amenable to automatic verification, using program analysis techniques (which trace their roots to optimizing compilers). Many challenges exist. The most notorious problems include the undecidability of the halting problem (which we overcome using approximations) and the need to deal with unknown input values. More recently, large codebases and plugin-based software architectures pose additional challenges to static analysis.

Nevertheless, a number of program analysis techniques for software engineering have been proposed and even deployed in commercial development environments. Facebook's Infer tool, Coverity, and Microsoft's Static Driver Verifier are three of the most successful examples of static analysis in practice.

In this seminar course, we will first briefly explore the strengths and limitations of program analysis techniques. These techniques traditionally come from the compiler research community. The bulk of this course, however, will consist of a discussion of current research papers in the field of software verification using program analysis techniques.

Objectives. After this course, you will:

- understand the strengths and limitations of static and dynamic analysis techniques;

- be familiar with the research literature on static and dynamic analysis for software verification.

- have carried out a small research project implementing a program analysis and evaluating its efficacy at solving a software engineering problem.

General Information

Course Web Page: `https://patricklam.ca/sase-2021`

Instructor:

> Prof. Patrick Lam
> Office Hours: By appointment
> Email: `patrick.lam@uwaterloo.ca`

Delivery: via videoconferences and discussion forums (Piazza), specific locations as above.

Course Description

Overview of techniques used in static and dynamic analysis, including dataflow analysis, type systems, and pointer analysis. Typestate properties. Applications to software engineering (notably concurrency and security).

Here is a brief week-by-week schedule:

1	Organization
2	Empirical Studies I
3	Empirical Studies II
4	Initialization Analyses
5	Dataflow Analysis
6	Pointer Analysis
7	Fuzzing
8	Testing
9	Repair
10	Domain-Specific Languages
11	Libraries

Reference Material

The reference material for this course consists of the research papers that we'll discuss every week. Most of the papers have been posted on the authors' webpages, and I've included those links when possible. You may need to use the campus proxy to download the ACM links to a couple of papers.

Evaluation

You will present one paper to the class and complete a course project, which includes a short presentation on the last day of class. There will also be a take-home final examination where you will demonstrate your understanding of the papers we've discussed as well as self-reflection on your course project.

Course project	50%	due last day of class
Project presentation	5%	during last class
Paper presentation	15%	throughout term
Final exam	30%	final exam period

Presentation. You will present one of the papers that we are discussing in this class by signing up on the Google Doc. Be prepared to answer questions on the papers.

I will provide submarks for both delivery and content of your paper and project presentations, and I'll send you timely feedback and suggestions on how to improve your presentations.

"Discussing" means that you need to be actively involved in the discussions of the papers. Typically classes like this one are small, so each of your contributions is important. You don't have to pipe in on every paper, but I expect you to contribute something substantial to at least one paper every two weeks. A contribution can be either a thoughtful question or a remark about a paper, and can be either live or on the discussion forum in the week of the paper presentation. (If you do not contribute to discussions, you will lose marks from the presentation category for your presentation.)

Course project. The course project gives you an opportunity to work on a particular application of program analysis techniques to software engineering issues. I expect that most projects will include an implementation component. To help you stay on track, I will expect a project proposal by the third week of class and a mid-term project update (1 page) by the end of week 8. (Start early!) I will post a list of suggested projects on the webpage. Please come see me to talk about possible projects!

Final examination. The open-notes final examination will ask you to synthesize the information you've seen throughout the semester. Potential questions include summarizing key points from selected papers, or sketching out how the techniques from one paper might apply to the problems that another paper addresses, as well as some self-reflection on your project.

Lateness. This is a graduate seminar, so I will offer some flexibility on due dates. However, this flexibility is often not in your best interest. You must consult with me before you hand in something late; otherwise, you will lose 5% on the project mark for a late project proposal or mid-term project report. The default mark for a late project submission is 0.

Recommended inclusion

Fair Contingencies for Emergency Remote Teaching. We are facing unusual and challenging times. To provide contingency for unforeseen circumstances, the instructor reserves the right to modify course topics and/or assessments and/or weight and/or deadlines with due notice to students. In the event of further challenges, the instructor will work with the Department/Faculty to find reasonable and fair solutions that respect rights and workloads of students, staff, and faculty.

Required inclusions