Static Analysis for Software Engineering (version 0)
ECE750-T5 (LEC 043), Spring 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. I am going to try to have some synchronous class-wide events, depending on where everyone is located, as well as synchronous smaller-group activities with subsets of the class, and one-on-one meetings. I'm confident that we can make that work.

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 to be provided later.

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).

A week-by-week schedule will appear on the course webpage by mid-December.

Reference Material

The reference material for this course consists of my notes for the first two lectures and 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 two papers 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.

I'm undecided about whether we should have pre-recorded or live paper presentations. A live presentation can take less time than a good pre-recorded presentation. We might try one of each. We will discuss this point and decide by the end of the first week of class, Friday January 15.

Course project	50%	due last day of class
Project presentation	5%	during last class
Paper presentations	10% each	throughout term
Final exam	25%	final exam period

Presentation. You will present two of the papers that we are discussing in this class. Please let me know which papers you have chosen by the end of the second week of classes. 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.)

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.

Required inclusions

Academic Integrity: In order to maintain a culture of academic integrity, members of the University of Waterloo community are expected to promote honesty, trust, fairness, respect and responsibility. [Check `www.uwaterloo.ca/academicintegrity/` for more information.]

Grievance: A student who believes that a decision affecting some aspect of his/her university life has been unfair or unreasonable may have grounds for initiating a grievance. Read Policy 70, Student Petitions and Grievances, Section 4, `www.adm.uwaterloo.ca/infosec/Policies/policy70.htm`. When in doubt please be certain to contact the department's administrative assistant who will provide further assistance.

Discipline: A student is expected to know what constitutes academic integrity [check `www.uwaterloo.ca/academicintegrity/`] to avoid committing an academic offence, and to take responsibility for his/her actions. A student who is unsure whether an action constitutes an offence, or who needs help in learning how to avoid offences (e.g., plagiarism, cheating) or about "rules" for group work/collaboration should seek guidance from the course instructor, academic advisor, or the undergraduate Associate Dean. For information on categories of offences and types of penalties, students should refer to Policy 71, Student Discipline, `www.adm.uwaterloo.ca/infosec/Policies/policy71.htm`. For typical penalties check Guidelines for the Assessment of Penalties, www.adm.uwaterloo.ca/infosec/guidelines/penaltyguidelines.htm.

Appeals: A decision made or penalty imposed under Policy 70 (Student Petitions and Grievances) (other than a petition) or Policy 71 (Student Discipline) may be appealed if there is a ground. A student who believes he/she has a ground for an appeal should refer to Policy 72 (Student Appeals) `www.adm.uwaterloo.ca/infosec/Policies/policy72.htm`. Note for Students with Disabilities: The Office for Persons with Disabilities (OPD), located in Needles Hall, Room 1132, collaborates with all academic departments to arrange appropriate accommodations for students with disabilities without compromising the academic integrity of the curriculum. If you require academic accommodations to lessen the impact of your disability, please register with the OPD at the beginning of each academic term.