CS7480 16319 Special Topics in Programming Languages: Advanced Program Analysis

SEC 01 - Fall 2016

Course Syllabus, Policies, and Schedule

Instructor

Prof. Frank Tip
www.franktip.org

Required Texts

There is no required textbook for this course. All materials will be provided by the instructor.

Course Prerequisites

There are no formal course requirements. However, students are expected to have MS level algorithm knowledge and MS level programming skills (e.g., as taught by NEU's bootcamp course). Some familiarity with compilation technology is helpful but not required.

Course Description

In the past decade, there have been great advances in the development of automated tools that help increase programmer productivity, by finding various kinds of quality problems in their code, and by assisting them with program understanding and maintenance. This includes tools for finding bugs and security vulnerabilities, test generation, fault detection and localization, etc. Many of these tools rely on program analysis to compute an approximation of a program's behavior. In this special topics course, we will study key publications in which static and dynamic program analysis algorithms are used to detect bugs and security vulnerabilities in programs, and how these algorithms are used in other tools that support programmers. Both theoretical properties and practical effectiveness of program analysis algorithms will be studied.

Specific topics that we expect to cover in this course include:

- static and dynamic techniques for finding errors, including type-based and dataflow-based techniques
- static and dynamic techniques for finding security vulnerabilities (e.g., taint analysis)
- automatic test generation (e.g., using dynamic symbolic execution)
- analysis challenges posed by dynamic and reflective programming language features
- specialized string analysis techniques for tracking the flow of string values in applications
- suitability of program analysis algorithms for different programming languages
- the use of open-source frameworks for program analysis

The course will be organized as follows. First, the instructor will review basic program analysis concepts and terminology (call graphs, points-to graphs, SSA form, etc.) and key applications of program analysis technology. Then, students and the instructor will present publications. Each paper presentation will be followed by a discussion. In addition to the presentations, students will do a modest-sized course projects, which could consist of, e.g., a literature study, the implementation/evaluation of a simple program analysis, or the evaluation/comparison of tools that implement a program analysis. Students will be expected to give a few short presentations about their project so that the other course participants can provide constructive feedback. Instead of a final exam, students will be required to write a short report about their project.

Course Outcomes

Upon completion of this course, students will be broadly familiar with program analysis and its applications. Through the project and presentations, they will have gained familiarity with the program analysis literature. They will understand the tradeoffs between precision and scalability, and the challenges associated with applying program analysis to programs written in real-world programming languages. In addition, they will gain valuable experience in giving technical presentations.

Research Paper Presentations

The students will take turns presenting research papers. Each lecture, one or two papers will be presented from a list provided by the instructor. Depending on enrollment numbers, each student should expect to present 2 or 3 papers during the course. The instructor will make an effort to align paper selection with each student's interests. Research paper presentations will be approximately 30 minutes in duration, to be followed by an interactive discussion of about 15 minutes. The instructor will occasionally give research paper presentations as well.

There is no required format for presentations, but an effort should be made to separate technical content of a paper (as presented by the authors) from the student's opinion/perspective of the work. Regarding technical content, the presentation should aim to cover the following:

- what problem is the paper aiming to solve?

    - why is that an important problem?
    - explain using a motivating example if possible
- outline of technical approach
    - explain algorithms (please illustrate using code examples, diagrams, etc.)
- results reported by the authors
    - has the work been implemented? any limitations? experimental results?
    - if a tool is available, please try it out and consider giving a demo!
- related work
    - how does the paper improve on previous work?
    - alternative approaches, applications, …
- conclusions

Furthermore, it is suggested that the student's includes the following to give his own perspective on the presented work:

- opinion of the work
    - technical comments (e.g., insights regarding precision, scalability)
    - limitations/problems (e.g., certain language features not handled)
- impact/positioning of the work
    - who has used the work, and for what?
    - has anyone improved on the work?
- 1-2 questions and/or topics to start an in-class discussion with

Reading

For each* lecture where a student does not give a research paper presentation, he/she is expected to read the presented papers in advance and prepare one or two questions. Students will be asked to pose some of their questions in class, following each presentation.

Note: providing summaries of papers presented by others is no longer required.

Policy Regarding Re-Use of Material for Research Paper Presentations

Given the nature of this course, where the focus is on understanding reviewing a large corpus of research literature, a lenient attitude will be adopted regarding the re-use of material. In particular, students are explicitly allowed to re-use examples and other content from the papers they present, provided that any such re-use is properly acknowledged. For some papers, students may be able to find existing presentations on the internet. Simply re-using an entire existing presentation is not allowed, but reusing examples, diagrams, charts is permitted provided that any reuse of such material is properly acknowledged. Students are warned that reuse of such "secondary" materials beyond the paper itself is at their own risk, in the sense that such materials may or may not accurately portray the work.

Course Project

Each student will participate in a course project. This project may be done individually, or in teams of size 2 (encouraged). Each team may select from the following options:

- implementation project
    - implement a program analysis and apply it to some subject programs
- evaluation/experience project
    - evaluate and/or compare existing program analysis infrastructure(s)
- literature survey project on a topic relevant to the course
    - should be comprehensive, well-organized, and self-contained
    - although your source papers may use different terminology and notation, your survey must be internally consistent throughout and must provide significant added value beyond the source material

Each project has the following associated deliverables: (i) a written final report (max. 10 pages), (ii) three short in-class project presentations as specified below, (iii) a copy of any developed software and subject programs. The in-class project presentations include:

- a 10-minute presentation about what you are planning to do, to receive feedback/suggestions from the class
- a 10-minute "status update" about the project mid-way through the semester
- a 10-minute final project presentation with lessons learned from your experience, conclusions, results, and a demo if possible.

Since the time available to work on the project is limited, it is necessary to restrict the scope of the work to fit the available time, particularly for implementation projects. This can be done by:

- relying on externally available front-ends/components/libraries where possible
- limiting yourself to analyzing a limited class of subject programs (e.g., programs that are fully contained in a single source file or module)
- ignoring certain features in the programming language being analyzed
- simulation/mock-up system of system components as needed
- teams are strongly encouraged to think of other ways to make the scope of their projects feasible

Teams are expected to document any such limitations in their reports and presentations, and are encouraged to discuss how some of these limitations could be eliminated. The goal of the course project is to learn about what is involved in developing a program analysis tool. The analysis does not need to be novel, and there is no expectation that a tool will be constructed that is ready to be used by others.

The final report about the course project will be due on Friday, December 9, 12pm eastern time. The report must be committed to each team's repository as PDF file named "report-teamN.pdf", where N is the number of the team. The final report should be 5-10 pages long using 10pt or 11pt font size. This includes all diagrams, references, figures, etc.

Assessment and Grading

This is a research-oriented special topics course where the focus is on presenting and understanding technical papers about program analysis, and on doing a small project to gain deeper experience with a selected topic or program analysis infrastructure. The grade for the course will be computed as a weighted average of scores obtained for the presentations and the project as indicated below. There will be no final exam.

- presentations: 50% (average score of research paper presentations)
- project: 50% (average of 3 project presentations and the final report)

Discussion and Communication

Discussion about course content, scheduling, general questions and answers will be handled using piazza (via the Blackboard distribution list if needed). Email exchanges with the instructor should be reserved for private communication and not for general questions about course content, materials, or papers.

Attendance

Attendance is mandatory for students who are scheduled to present about papers or about their project. The instructor is currently not planning to enforce attendance for non-presenters, but students are reminded that much of the value of this class will be in the interactive discussions following the presentations. Please provide an engaged audience for your fellow students, and they will return the favor when it is your time to present!

Submission of Work

All work for the course is expected to be completed by the due date and time. The CCIS GitHub installation at https://github.ccs.neu.edu will be used for submitting PDF/Powerpoint of paper presentations and project-related materials. Further details on submission procedures will be announced in class.

Accommodations For Students With Disabilities

If a student has a disability-related need for reasonable academic accommodations in this course and has not yet met with a Disability Specialist, then visit www.northeastern.edu/drc and follow the outlined procedure to request services.

If the Disability Resource Center has formally approved an academic accommodation in this class, the student must present the instructor with a “Professor Notification Letter” during the first week of the semester, so that the instructor can address specific needs as early as possible.

Academic Integrity Policy

The University views academic dishonesty as one of the most serious offenses that a student can commit while in college and imposes appropriate punitive sanctions on violators.

Students are expected to read and understand the Northeastern University Academic Honesty Policy found here. In general, unauthorized collaboration is any collaboration that has not been specifically authorized. However, in this course we specifically list any form of file sharing as unauthorized.

Any form of cheating or sharing of files or assignments (whether receiver or provider) will result in a grade of 0 for that assignment, a report to OSCCR, and a full two letter reduction in the final grade. These penalties apply for each instance of cheating or file sharing.

Title IX

Title IX makes it clear that violence and harassment based on sex and gender are Civil Rights offenses subject to the same kinds of accountability and the same kinds of support applied to offenses against other protected categories such as race, national origin, etc. If you or someone you know has been harassed or assaulted, you can find the appropriate resources here: http://www.northeastern.edu/oidi/titleix/.

No-Shows and Late Submissions

0 points will be given when students do not show up in class on days when they are scheduled to present because such non-attendance would be highly disruptive. Exceptions to this rule will only be granted for a valid medical excuse accompanied by a doctor's note.

For the final project, 10 percentage points will be deducted for every day that it is submitted late.

Schedule and Reading List

The course schedule and reading list can be found here.

Lecture Materials

Lecture-9Sep2016
Lecture-13Sep2016