Instructor: Wei Le (weile@iastate.edu)
Class schedule: M W 11:00am-12:15pm, Black 1026
Office hours: Upon request (your email title should include 513x; in the email, please include a few time options as well as the subject for the meeting.)

1 Course Description
Program analysis aims to automatically reason about the code and program executions for predicting software behaviors. The course will teach how to determine program properties (foundational knowledge of program analysis) and what are the purposes to do so (applications of program analysis). The course will teach concepts, algorithms, tools, benchmarks and methodologies needed for solving problems using program analysis and for establishing foundations for doing research in program analysis.

2 Course Objectives
After successfully completing this course, students will be able to:

1. know terminologies and mathematical frameworks needed to read program analysis literature,
2. learn and implement classical program analysis algorithms (you are encouraged to open source your implementation),
3. understand the connections between program analysis and other fields such as software engineering, security, programming languages, machine learning, artificial intelligence and statistics, and
4. formulate software problems to program analysis and apply program analysis tools to solve them.

3 Prerequisites
• COM S 331: Theory of Computation
• COM S 342: Principles of Programming Languages

4 Textbooks and Other Resources
The course does not have a required text book. We will use lecture notes and papers to teach and learn, though the following books are considered classical to learn program analysis:

• *Advanced Compiler Design and Implementation* by Steven Muchnick, published by Morgan Kaufmann, ISBN 9781558603202: it is a compiler book that covers the topics of control flow analysis, dataflow analysis, alias analysis, and the applications of program analysis in compiler optimizations.

You are also welcomed to check out the program analysis courses taught by other instructors:

• Alex Aiken, Stanford, [CS 357 Techniques for Program Analysis and Verification](#)
• Monica Lam, Stanford, [CS243 Program Analysis and Optimization](#)
• Jonathan Aldrich, CMU, [15-819 O Program Analysis](#)
• Jens Palsberg, UCLA, [CS232 Static Program Analysis](#)
• Mayur Naik, Georgia Tech, [CS6340 Software Analysis and Testing](#)
• Stephen Chong, Harvard, [CS252r Advanced Topics in Programming Languages](#)
• Mooly Sagiv, Tel Aviv University, [Program Analysis](#)
• Evan Chang, University of Colorado Boulder, [CSCI7135 Program Analysis: Theory and Practice](#)

5 Course Work

• Survey (30%): You will team with another student in class to write a 6-page survey (including the references) on the topic of interest (tentative midterm week, March 2nd).

• Assignments (40%): You will finish four assignments (10% each) related to implementation of program analysis or using the tools of program analysis.

• Research project (30%): You will team with another student to accomplish a research project on the topic of program analysis. The submissions include a proposal (tentative at the end of the first month), a mid-point check presentation (tentative at the week after spring break, March 19th), and a final report with a demo (final exam time, May 1st).

6 Tentative Topics

A typical class will consist of lecture and discussion time:

Lecture topics, including but not limited to:

1. Theoretical complexity of program analysis
2. Abstract interpretation: theory of abstraction, soundness and completeness
3. Control flow analysis: predicting execution paths
4. Data flow analysis: predicting data access patterns (definition, use, dependencies)
5. Chop, slice, dice: extension of data flow analysis
6. Taint analysis: extension of data flow analysis
7. Value flow analysis: predicting which expressions produce the same values
8. Pointer analysis: predicting which pointers points to the same memory location (value flow of the pointers)

9. Symbolic execution: representing program states using symbols (sometimes symbols mixed with concrete values)

10. Interprocedural analysis: scope of the analysis

**Open questions to discuss, including but not limited to:**

1. Which program analysis frameworks to use?

2. Analyzing binary level, source level or intermediate level of the code?

3. What problems are solved using program analysis?

4. What other techniques are used with program analysis?

5. What are the future directions of program analysis?

6. What are the mathematical problems the program analysis problems are reduced to? For example, points-to analysis is reduced to reachability on graphs.

7. What are the program analysis challenges for different programming languages, e.g., C, Python and Java?

8. A key spirit of static analysis – abstraction and approximation

9. What are the models and representations of programs?

10. What are the popular program analysis tools and why they are successful?

### 7 Course Policies

**Late homework:** We do not grade late homework. Please submit your homework in time.

**Academic dishonesty:** Suspected academic misconduct will be reported to the dean of students office [http://www.dso.iastate.edu/ja/academic/misconduct.html](http://www.dso.iastate.edu/ja/academic/misconduct.html)