Course Project

17-355/17-665/17-819: Program Analysis
Rohan Padhye

Due dates listed below; 400 points total

An overarching goal of this course is to provide students the expertise necessary to (A) understand, appreciate, criticize, and reuse key ideas in program analysis, and (B) put those ideas into practice in state-of-the-art analysis tools. The goal of the project is to become deeply familiar with a particular subarea of program analysis research or practice, and to demonstrate that familiarity through a research or engineering-focused course project. The expected scope of the project is about 30–40 hours per person, spread out over the last month of the course (note that we have no final exam).

Project options

This project has two flavors: the Research variant and the Practice variant. Research is required for students enrolled in the Ph.D. version of the course, 17-819, and optional for others. Undergraduate and master's students taking 17-355 or 17-665 may elect to do research; please note in the project proposal if you want to do this. Additionally, students enrolled in 17-665, the master's version of the course, must engage with nontrivial codebases as part of their projects. Either the analysis framework or the target program must be in active use by the developer community.

You have many options, regardless of whether you aim for a research or practice-oriented project. For the *practice* variant, you might:

- Design and implement a non-trivial, interesting analysis. Typically we will expect this to target a real programming language. However, if you are interested in meaningfully extending the WHILE and WHILE3ADDR compiler/interpreter with a non-trivial language feature and associated analysis, let us know and we will help ensure you've identified a task of reasonable scope.
- Carry out an empirical evaluation of one or more existing analysis method(s) and/or open-source tool(s) (typically 3–5 such tools).
- Contribute meaningfully to an open-source static analysis tool.

For the *research* variant, PhD students (at least) must:

- Either propose a new (modest) program analysis research project, or (more likely) conduct an extension of your existing research, so long as it is new and includes a non-trivial analysis component. Many students in the course already study, use, or extend various analyses in their current research.[1] It is therefore acceptable to integrate/make use of your current

[1] Recall that analysis includes static and dynamic techniques, and not simply the material we covered in the course so far.

research for the project, but you must plan, make, and document novel progress. That is, you may not simply write up analyses and results you already have. In the proposal, provide sufficient background on your current project that we can understand the high level goal, identify the components that relate to program analysis, and describe the scientific questions you will investigate; how you will investigate them (empirically, theoretically, etc.); and how you will evaluate. Sketch the initial ideas that underlie your solution or hypothesis. Check with us if you are not sure whether your research qualifies.

Undergraduate or Masters students are also welcome to do real research following the above lines. If you are not sure what to do but you have an area of interest, contact the course staff and we can help you refine your interests. Alternatively, if you are an undergraduate/masters student interested in research you can instead:

- Critique state-of-the-art research in a program analysis subject of our choice by reading, analyzing, and comparing a set of research papers (three is the expected number) in that area in a detailed essay.

If you have an idea for something you want to do that somehow doesn't fit in any of the above options, contact us! We are happy to work with you to find something that fits with your interests.

We provide additional guidance for several of the above options in the Appendix, below.

Deliverables

The project consists of several deliverables, each with their own due dates, all submitted via Gradescope.

See the Appendix for more specific guidance, per-project-type, for each of these deliverables.

- **Proposal:** *(25 points)* Thursday, November 13, 2025, 11:59 pm).
- **Checkpoint:** *(25 points)* Friday, November 21, 2025, 11:59 pm
- **Presentation:** *(150 points)* Thursday, December 11, 2025, 1:00pm-4:00pm
- **Report:** *(200 points)* Thursday, December 11, 2025, 11:59 pm. No late days permitted.

Solo vs. Teams. Students may work individually or in groups of up to two, of their own selection. PhD students working on thesis-related topics are usually expected to work on their own unless a class-mate is also a research collaborator / paper co-author, in which case we will allow group-work. Group projects will be given a single grade, with exceptions only in extraordinary circumstances. Expectations for a group project will be higher than for a solo project.

Project scope/proposal. *(Thursday, November 13; 25 points)* The first deliverable is description of what you intend to do for your project. This description can be brief (1–2 paragraphs). List the project type, title and the members of your group (or indicate that you are working alone). Outline what you are going to do, why it is interesting, and how you are going to evaluate it, as well as what, roughly, you plan to do do for the checkpoint as a means to remove any uncertainty or risks in your initial project goals (e.g., run something on toy examples, or implement a sub-set of your overall analysis to get used to the framework and understand its limitations).

Checkpoint. *(Friday, November 21; 25 points)*. Submit a short status update that describes what you have done, if you are on schedule with respect to your original plan, and any issues or concerns with the project. If all is going well, a short paragraph will suffice. Include evidence (code, data, design document) of your work so far. There are no requirements on the form of the evidence, i.e., it doesn't have to compile or run. This is the last opportunity to tweak your project goals if you

need to. After this point you will be graded on how well you execute the project based on your outlook at the time of your checkpoint.

Presentation. *(Thursday, December 11, 1:00pm-4:00pm; 150 points)*. Prepare a 6–8 minute presentation (you will be cut-off at 8 minutes!!) describing your project for the benefit of your peers. Describe the problem you are solving and why it is important; give high-level background not already provided in the class; explain how you solved the problem, and show an indicative result or short demo, depending on the contribution of your project. Focus on identifying a cool thing you did or learned, and showing or explaining that cool thing to your classmates. We will structure the final exam slot to watch the presentations and ask questions/interact as a class, and interaction/questions will be considered in final grading as much as the format allows. Please upload your slides to Gradescope before the presentation slot, as we will use this submission for grading.

Report. *(Thursday, December 11; 200 points)*. Turn in a report describing what you did, why, your results, and reproduction steps (or links to pull requests!) as necessary. We are flexible on length so long as you do all that is necessary to convey the information requested: fewer than around 4 pages is probably too short; more than 8 is almost certainly too long for most projects (with possible exceptions for research-oriented projects). Include information necessary to understand and reproduce what you did and your results. All scientific results, as well as input data, should be included or referenced. If your project involved implementation, submit a link to the source code and necessary documentation for understanding or running it, as appropriate.

Appendix

In this section, we provide some additional guidance on what we expect in the various deliverables based on the type of project you propose.

Nontrivial analysis, empirical evaluation/comparison

For the proposal, what type of analysis will you implement/compare, for what language, and using what framework(s)? How will you evaluate success (at a high level, i.e., what's your general test strategy), and/or what metrics or approach will you be using to compare frameworks, methods, or tools? You may be weighing different options, still; if so, be concrete about what those options are and how you will choose between them. E.g., if you want to target C, you may want to choose between CIL or Clang as a framework (or others...). You need not decide in the proposal, but you *do* need to at least list what options you are considering.

For the checkpoint, concretize any decisions you did not make in the proposal, provide evidence of progress in the implementation direction (e.g., evidence that you have compiled a toy analysis in the framework if it is new to you, or that you have set up baseline or simple testing of analysis or gotten various comparative tools to compile or run on a simple test case).

Open Source Contribution

There exists a large number of open-source analysis tools on GitHub; you might consult the GitHub Marketplace "Code Quality" tab, or `https://github.com/analysis-tools-dev/static-analysis`, which lists a number of tools (they also have a page for Dynamic Analysis) that target a variety of languages.

If you choose this project type, your goal is to select an open-source project and complete one or more bug fixes or extensions within it. You have considerable freedom in which project and tasks you choose. In general, you are more likely to succeed if the project is active and has multiple contributors. Beyond that:

- You *must* choose a bug report or feature request from a public database or message board, following whatever protocol the project uses to communicate and track open issues. Do not invent a task; Address an actual, documented project need.
- The task must require changes to the project's source code. Pure documentation or design tasks are not appropriate.
- You may choose one large task or several smaller, related tasks.

Choosing a task of the appropriate size can be challenging; in estimating, bear in mind that it takes time to get the lay of the land with a new project.

For the proposal, ideally you will be able to list both the project and 1–3 candidate tasks you hope to take on. At minimum, you should give some options of projects/tasks so we can smell check the scope for you.

For the checkpoint, concretize any open questions from the proposal in terms of project/tasks, and provide evidence of progress (such as compiling the program and identifying tests that help you understand the needed feature, or the module where the change will take place).

For the report/presentation, be sure to explain at least what the project does as well as necessary context to understand the nature of the task you took on. So long as you aren't "scooped" on the task, we do expect you to *submit your changes to the project via pull request*. This may involve adhering to project-specific submission and code quality guidelines, or communicating with members of the project community to help support acceptance. If your pull request is accepted, you will receive extra credit.

Research

You have considerable leeway on all elements of the project with respect to current research. Please be clear in your proposal what you are doing, how it relates to your current research (if applicable), and what the analysis angle is. Further be clear about your checkpoint. We expect the report to be on the longer side, and to have a fairly detailed related work section.

If you would like to do a research project but don't know where to start, contact us ahead of the proposal deadline and we will work with you.

Critique[2]

If you take on a research critique, your goal will be to conduct a deeper dive into a particular area of analysis. You will read at least 3 papers in the area, and write a detailed essay (a "critique") that discusses and compares the approaches taken in the three papers. You should aim to analyze three research papers that solve the same problem in different ways, and then compare and contrast the different approaches, discussing what the papers share in common, what is different across them, and to what extent those differences are complementary, orthogonal, or mutually exclusive to each other.

You will be choosing the set of three research papers that you analyze yourself, subject to approval by the course staff, as well as these constraints:

- At least one paper from your set must have appeared within the past five years (the other two can be older than that, if necessary).

[2]This material is adapted from an assignment from 15-300.

- The papers must be full-length conference or journal papers, which are typically 10 or more pages long. (They cannot be "short papers" that are just a few pages long).
- None of the papers should be "survey" papers (which summarize a research area, but do not focus on presenting new approaches to solving the problems).
- Your set of papers must be approved by the course staff.

In the proposal, you should at least identify the area in which you would like to read papers, and at least one paper you will start with. If you would like to do this option but are flummoxed, reach out to us before the proposal deadline and we will help you.

For the checkpoint, you should have selected all three papers. List them, with citations, and a sentence that explains the ways in which they are related. You will likely have to look over/skim more than three papers to find a good set (though you need not read them in excruciating detail).

Some tips:

- Perhaps the most useful starting point for finding papers that are closely related to a given paper is to look at the list of papers that it cites. A research paper is supposed to cite (and discuss) previous work that is closely related. The list of citations appears at the end of the paper (usually under the heading, "References"), and the most relevant citations are often discussed within the paper (sometimes under an explicit "Related Work" section).
- While citations are helpful in identifying related work that was published a year or more before a given paper, what about related papers that were published either concurrently with or after a given paper? One way to identify closely-related conference papers that appeared concurrently with each other is to look at the session in which they were published at the conference: conferences are usually organized thematically into sessions with particular themes, and there is a good chance that closely-related papers will end up within the same session.
- It is also helpful to identify papers that appeared later in time that cite a given paper. Several of the major web sites that allow you to search for CS research papers (e.g., the ACM Digital Library (`http://dl.acm.org`), Google Scholar (`http://scholar.google.com`), etc.) include a "Cited By" feature.
- Finally, another very helpful resource for identifying sets of closely-related research papers is a PhD student who is working in that area. PhD students are usually quite familiar with all of the related work in their area of study, or the course staff. Feel free to reach out to such students, or to us, for pointers.

For the presentation, be sure to at least explain the high-level problem the three papers are solving, and key similarities/differences in the approaches. You don't have time to go into incredible detail on all three; focus on the most important or interesting features of each approach.

The bulk of the report should focus on describing and comparing the three research papers. First describe the "big picture": what problem are they trying to solve, why is this problem important and challenging, and any relevant background. Then, describe each of the three papers: What approach did they take to solve the research problem? What were their key contributions? What were their limitations? Finally, you will also contrast the three papers: What did they share in common? What were the major differences between the papers? Regarding these differences, were they complementary, orthogonal, or mutually exclusive to each other? Conclude as you see fit, but one good approach might be to identify questions that remain open, limitations of all of the approaches, or a judgement about which approach is most promising for future application or research.