02242: Project Description

Christian Gram Kalhauge / DTU Compute

The goal of this project is to demonstrate that you have achieved the learning objectives of the course. To reiterate, a student who has met the objectives of the course will be able to:

- explain the concepts of soundness and completeness and the limitations of program analysis;
- explain the benefits and tradeoffs associated with syntactic, semantic, dynamic and static analyses;
- design an analysis to solve a specific problem;
- formulate theoretical guarantees and limitations of the analysis;
- argument for the approach in a project proposal;
- implement an analysis for part of a real programming language;
- design and motivate a series of experiments using this analysis and interpret the results obtained;
- communicate their results in a clear and precise manner in a paper format; and
- achieve the above goals in a group effort while at the same time maintaining individual accountability.

To effectively test all of this, you as the students in groups of four, will work on first crafting a proposal of the project, and then carrying out the project. Handing in the results as a 10-page paper, which is defended orally.

Timeline

The project has the following timeline:

Date	Topic
`10/23`	Last day for handing in project proposals.
`11/23`	Final paper deadline.
`12/07` + `12/08`	Oral Exam.

Formats

WARNING: Deviating from the formatting requirements may result in a automated failed grade

Both the proposal and the final paper MUST be formatted according to the IEEE - Manuscript Templates for Conference Proceedings. For maximal ease, use the Overleaf Template. We strongly recommend that you use LaTeX to format your paper. The proposal can be no longer than 2 pages (excluding references), and the paper can be no longer than 10 pages. All responsible authors of the paper should be listed with student IDs. Only admissions in PDF are accepted.

The Project

The project should solve a problem, either with an application of a program analysis or by an implementation an analysis technique. Note, that this course defines program analysis broadly as extracting information from a program. The project can analyze any programming language, and use any technique you find interesting, however, we expect you to use good academic judgement and show fulfillment of the learning objectives.

The Proposal

The proposal should be the introduction of the paper (see next section), but instead of stating your solution, you should state your planned solution as well as hypotheses. On top of the proposal you should also include a week by week plan for how to achieve the goals set out in the proposal, and an evaluation strategy.

- Remember, 4 weeks are not a lot of time, you should have a good idea of where you are going, when you propose the project.
- Experiment early, even before handing in the proposal, so that you know the risks of the project.
- Do not wait for approval before starting the project, but be ready to pivot.

The Paper

First! Consider the advice in this lecture, by Simon Peyton Jones.

The final paper should contain the following sections (approximate page numbers in parentheses):

- An abstract, describing the paper.
- An introduction (1-2 pages), with
  - a good motivation and description of the problem you are trying to solve (consider using examples),
  - the problem stated as a research question,
    - e.g., “Is pentagons better than bound at approximating the state space of our examples?”
  - a short description of your solution, and
  - a list of contributions with forward references, described in the lecture by Simon P. Jones.
- A high level description of the technique used to analyze (2-4 pages), with
  - a formal description of the approach;
  - a walkthrough of the theory needed to understand the technique; and
  - any theoretical guarantees of the approach.
- A description the implementation and setup (1/2 - 1 page), with
  - a link to your source code; and
  - a high level description of the code.
- A describing the evaluation and the results (2-3 pages), with
  - an empirical evaluation of the approach,
  - compared against at least one other technique (or variations (optimizations) of your technique),
  - an analysis of the results, and
  - a discussion of the threats to validity.
- A discussion of the technique, and alternatives techniques to solve the same problem (1 - 2 pages), with
  - an explanation why other techniques form the lectures (and potentially the literature) were not used, and
  - a discussion of the good and the bad things about your approach.
- A conclusion (0 - 1/2 page), which is
  - A short recap of the paper.

Project Ideas

You are free to choose your own project, but here are some ideas:

Not all of these projects are completely doable in 4 weeks, choose with caution!

- Build a tool that can make detailed sequence diagrams from executions.
- Implement a concolic executor in Python, by dynamically overloading the `+` and `-` operations of most objects.
- Implement Pentagons in your current framework (Logozzo and Fähndrich 2008).
- Check if a java program contains null-pointers, using the `@NotNull` and `@Nullable` annotations: NotNull (Java(TM) EE 7 Specification APIs).
- Build a decompiler from Java byte code.
- Build an extremely efficient static analysis, by optimizing the analysis algorithm.
- Build a static analysis that works for method calls and classes as well.
- Interpret, analyze, and optimize brainfuck code.
- Build a tool that can predict which tests to rerun given a change in a program.
- Build a tool that can predict the semantic versioning of a piece of code.

Errata

- Add evaluation strategy to project proposal.

Logozzo, Francesco, and Manuel Fähndrich. 2008. “Pentagons: A Weakly Relational Abstract Domain for the Efficient Validation of Array Accesses.” In *Proceedings of the 2008 ACM Symposium on Applied Computing*, 184–88.