Introduction to CSCI 3294

Program Analysis
Dr. Mark Lewis
9-2-2001

Personal Introduction

- Education
  - 1996 BS in Computer Science and BS in Physics from Trinity
  - 1998 MS in Astrophysics and Planetary Science
  - 2000 MS in Computer Science
  - 2001 Ph.D. in Astrophysics and Planetary Science
- Extracurricular activities: basketball and volleyball

Research Interests

- Dissertation was on "Collisional Dynamics of Strongly Perturbed Planetary Rings"
- I have research interests in Programming Languages Theory, Parallel Computing, Quantum Computing, and Large Scale Simulation.

Syllabus Overview

- Course Webpage - http://www.cs.trinity.edu/~mlewis/CSCI3294-F01
- Contact info - Office HAS 201K, Phone 999-7022, e-mail mlewis@trinity.edu
- Office hours - 9:30-10:00 MWF, 8:30-11:30pm T, 3:00-4:00pm MW or by appointment
- Text - None. The readings for this course will be research papers from journals and proceedings.

Course Description

- In this class we will look at various topics in program analysis. Most of the semester will be focussed on pointer or points-to analysis. This is the process of figuring out what allocation sites the pointers in a program are able to point to. At the end of the semester we will see some of how this is applied so that compilers can optimize generated code. We will also have at least one class period to cover a paper on a topic of the classes choosing.

Class Format

- For the first five classes this course will be run as a normal lecture format class though perhaps with a bit more discussion than many of your other classes.
- Beyond that the class will be "run" by you as you give presentations on papers. Each day 2-3 students will present the paper for that day so that in the end each of you will have made two presentations.

More on Format

- While only 2-3 students will be presenting on a given day, the entire class is expected to have read every paper so that we can engage in discussions. I will also want each student to come to class with some written questions about the paper. Those will be used to start the discussion after the normal presentations are done.

Grading

- The course grade will be based on three components: your presentations (60%), your participation in class when not presenting (20%), and a final exam (20%).
- The final exam will be open note and open "book" so you can bring basically any reference you want into it. I don't intend it to be that hard, it's just to make sure you have understood what we have talked about to a reasonable degree.

Schedule of Papers

- The syllabus on the web includes a listing of papers and a general description of what each one deals with. I need you to pick which papers you are interested in presenting so that I can add you to that page. I will be doing the first five weeks of class to help bring you up to speed on topics that will be important for you to understand the readings.

What do you know now?

- How many of you have taken the Principals of Programming Languages class?
- Survey of languages
  - What languages do you know?
  - What are the strengths and weaknesses of each?
  - What paradigm does each one generally follow?

What is Program Analysis?

- The name says it all here, this includes anything that one might want to do to try to gather more information about a program for various purposes.
- What types of things can you think of that we might want to know about a program? Why would we want to know those things?

What is Points-to Analysis?

- Points-to analysis is one of the most significant forms of program analysis.
- It gives us information on what memory locations a specific pointer can point to.
- Different types of points-to analysis give us more or less information.
  - Context sensitive analysis
  - Flow sensitive analysis

When do we use them?

- The most obvious time to use program analysis is probably as part of a compiler when extra information about the nature of the program can aid optimization.
- In addition to that program analysis can be used both at compile time and run time to help track down bugs in programs.

Uses for Optimizing

- Removing virtual function calls when only one subclass can reach a particular call.
- Identifying natural parallelism when section of memory are independent.
- Putting objects on the stack or removing semaphore checks in Java.
- Determining memory use to intelligently implement preloading of memory.

Impact of Newer Hardware

- Some of these optimizations would not have been important 5 years ago. However, the nature of processors and the location of bottlenecks has changed in that time.
  - The gap between processor speed and memory latency has increased so cache memory is much more significant.
  - Processors have to have extended pipelines to reach higher clock rates.

Uses for General Debugging

- Language choices like strong typing can allow for better error detection at compile time and analysis can strengthen this further to find likely errors before the program ever runs.
- Bugs that get through can be more easily isolated with programming slicing and analyses that look at effect propagation.

Minute Paper

- In most of my classes I do this every class meeting. I will only do it the first 5 weeks in this class.
- What things that I've discussed seem most interesting/appealing to you? What do you think I will need to cover in the next 4 weeks so that you are prepared to read the papers you will be presenting?