Program Analysis: Theory and Practice

Evan Chang

Meeting 1: Welcome

CSCT 7135, Fall 2010
http://www.cs.colorado.edu/~bec/courses/csci7135-f10/

Introductions

• Who am I?
• About you?
  - What do you want to get out of this class?

Administrivia

• Website
  http://www.cs.colorado.edu/~bec/courses/csci7135-f10/
  - readings, slides, assignments, etc.
• Moodle
  - discussion forums, assignment submission
• Office hours
  - MW 5:15-6:15
  - and by appointment
  - ECOT 621

Today

• Some context and motivation
• Program analysis in a nutshell
• Goals for this course
• Requirements and grading
• Course summary

  • Convince you that program analysis is cool and exciting!

Meta-Level Information

• Please interrupt at any time!
• It's completely ok to say:
  - I don't understand. Please say it another way.
  - Slow down! Wait, I want to read that!
• Discussion, not lecture

No Useless Memorization

• I will not waste your time with useless memorization
• This course will cover complex subjects
• I will teach their details to help you understand them the first time
• But you will never have to memorize anything low-level
• Rather, learn to apply broad concepts
Discussion: What is Program Analysis?

- Checking for bugs (Sark)
- Examining source code (Sam)
- Traces/executions (Robby)
- Optimizations (Han Chao)
- "Automated reasoning/examination of program artifacts"

What is Program Analysis?

- Static Program Analysis?
  - Look: Predict what will happen
- Dynamic Program Analysis?
  - Trace: Observe what is happening

Is Program Analysis Useful?

- How is it used?
  - Compiler optimizations
  - Verification - proving bugs don't exist
  - Type checking/maintenance (a kind of static program analysis)

Is Program Analysis Useful?

- How is it used?

Big Motivation: Software Quality

* Comics: www.phdcomics.com
Big Motivation: Software Quality

Windows “Blue Screen of Death”

Microsoft uses and distributes the Static Driver Verifier

Airbus applies the Astrée Static Analyzer

Companies, such as Coverity and Fortify, market static source code analysis tools

But there’s hope in program analysis

Because program analysis can eliminate entire classes of bugs

For example,
- Reading from a closed file: `read( );` ❌
- Reacquiring a locked lock: `acquire( );` ❌

How?
- Systematically examine the program
- Simulate running program on “all inputs”
- “Automated code review”

Program analysis by example: Checking for double acquires

Simulate running program on “all inputs”
```c
... code ...
// x now points to an unlocked lock
acquire(x);
... code ...
```

Simulate running program on “all inputs”
```c
... code ...
// x now points to an unlocked lock in a linked list
acquire(x);
... code ...
```
Uh oh!

• Rice’s Theorem: “Any non-trivial property of programs is undecidable”

Should we go home then?

Abstract!

... code ...

// x now points to an unlocked lock in a linked list

acquire(x);

... code ...

Abstract Interpretation

[Cousot and Cousot 1977]

Abstractions

• Examples?

\[
\begin{align*}
\text{abstract: } & \mathbb{Z} \rightarrow \{ -, 0, +3 \} \\
\text{constant properties: } & x = 3 \rightarrow x = 3 \\
\text{by: } & x \rightarrow y = x
\end{align*}
\]

Challenges in Designing Static Analyses

Undecidability → Approximation

Soundness = Correctness

Precision < Performance / Resource Usage

Scalability ≠ Usability
Goals

Goal 1
Learn to build program analyses

Goal 2
Grok the theory behind correct program analyses

Goal 3
Understand current PL research (POPL, PLDI, OOPSLA, TOPLAS, ...)

Most Important Goal
Have Lots of Fun!

Requirements
Prerequisites

- "Programming experience"
  - building analyses in OCaml
  - ideal: undergraduate compilers (e.g., CSCI 4555)

- "Mathematical maturity"
  - basic logic
  - ideal: programming language semantics (e.g., CSCI 5535)

- Not strict. If concerned, please see me.

Assignments

- Reading and participation (each meeting)
- Homeworks
- (Possible) Presentation of Papers
- Final project

Reading and Participation

- ~2 papers/book chapter, each meeting
  - Spark class discussion, post/bring questions

- Online discussion
  - Post ≥1 substantive comment, question, or answer for each lecture
  - On csci7135-f10.blogspot.com
  - Due before the next meeting

What is “substantive”?

- May be less than a standard blog post but more than a tweet.
- Some examples:
  - Questions
  - Thoughtful answers
  - Clarification of some point
  - What you think is the main point in the reading set.
  - An idea of how some work could be improved
  - Comments on a related web resource related

- Intent: take a moment to reflect on the day’s reading/discussion (not to go scour the web)

Homework

- Homework
  - First half of the semester only
  - Collaborate with peers (but acknowledge!)

Final Project

- Implement an advanced abstract domain and apply your analysis
- Write a ~5-8 page paper (conference-like)
- Give a ~15-20 minute presentation
Academic Dishonesty

- Don’t do it!
- See CU Honor Code

Course Summary

Course At-A-Glance

- Part I: Program Analysis Foundations
  - Interleave implementation and theory
  - Textbook:
    - Other notes and papers
- Part II: Research Applications

Possible Special Topics

- Software model checking
- Shape analysis
- Symbolic/concolic execution
- What do you want to hear about?

Next Time

- Guest lecture by Xavier Rival on the Astrée Static Analyzer
- Read the paper on Astrée
  - see the website under “Schedule”
- Also: Overview Talk, Tue 11-12, ECOT 831

Next Mon: Model Checking

- Verify properties or find bugs in software
- Take an important program (e.g., a device driver)
- Merge it with a property (e.g., no deadlocks)
- Transform the result into a boolean program
- Use a model checker to exhaustively explore the resulting state space
  - Result 1: program provably satisfies property
  - Result 2: program violates property "right here on line 92,376!"
Remember

- Join the course moodle and introduce yourself on the blog
  - Write a few sentences on why you are taking this course