Static Analysis:
Automated Bug Hunting and Beyond

Julian Erhard   Michael Schwarz
{julian.erhard, m.schwarz}@tum.de

Chair for Formal Languages, Compiler Construction, Software Construction
Department of Informatics
Technical University of Munich

Summer Term 2022

Writing programs is hard.

Writing correct programs is very hard.

Testing

- Widely successful
- Can be automated to some extent
- Can only show that there are bugs, not their absence

Machine-verified proof (e.g. Isabelle)

- Can show bugs & their absence
- A highly manual process requiring highly trained people
- Problem with proof and implementation diverging

Static Analysis

- Fully automated
- Can show absence of certain classes of bugs
- Runs directly on the input program
- Abstract Interpretation, Model Checking, ...

Static Analysis

- Fully automated
- Can show absence of certain classes of bugs
- Runs directly on the input program
- Abstract Interpretation, Model Checking, …

Abstract Interpretation

- Widely used both in Academia & Industry
- Can scale to huge industry-scale codebases
- The technique covered in Program Optimization Course (IN2053)

Goblint

- Analysis of multi-threaded, real-world C
- Efficient solvers for computation of fixpoints
- `https://goblint.in.tum.de`

Topics

- Abstract domain for **floating point** numbers
    - Important part of many programs, especially embedded
    - We have various domains for integers, but none for floats

Topics

- Abstract domain for **floating point** numbers
  - Important part of many programs, especially embedded
  - We have various domains for integers, but none for floats
- More expressive integer domains for **detection of overflows**
  - Integer overflow for signed types is undefined behavior in C
  - e.g. Interval Sets

Topics

- Abstract domain for **floating point** numbers
  - Important part of many programs, especially embedded
  - We have various domains for integers, but none for floats
- More expressive integer domains for **detection of overflows**
  - Integer overflow for signed types is undefined behavior in C
  - e.g. Interval Sets
- **Tooling** surrounding Goblint
  - Present analysis results to developers / users
  - Web-based frontend leveraging `Js_of_ocaml`

Benefits

- Deepen your understanding of
  - The Semantics of C and typical programming errors
  - Static Analysis by Abstract Interpretation
- Train your functional programming skills
- Give some insights into developing a research prototype

Format

- Teams of 2-4 students
- Course will take place throughout the semester
- (Bi-)weekly meetings with us, default in person
- Presentation at the end (one day, all groups)
    - Attendance & Active Participation mandatory(!)

Requirements

- Program Optimization Course (IN2053)
- Knowledge of a functional programming language (we use OCaml)
- Be in your Master's (Advanced Bachelor's students welcome)

Questions?