Automatic Program Analysis

WWW: http://www.cs.uu.nl/docs/vakken/mapa/
E-mail: i.g.dewolff@uu.nl

Edition 2021/2022
Course overview
What is automatic program analysis about?

▶ A semantics-based, static approach to the analysis of program artefacts (i.e. the source code)
▶ Generally, it can be much broader than that:
  ▶ dynamic analysis
  ▶ hybrid analysis
  ▶ software comprehension
Why do people study static analysis?

Static analysis is a crucial tool in

▶ program optimization
  ▶ Which statements will never be executed?
▶ program validation
  ▶ Is this program type correct?
▶ program understanding (comprehension)
  ▶ What is the architecture of a 5 mln. Cobol system?
  ▶ Not the focus of this course. Maybe a lecture at the end.

Basic ingredients also useful in other settings.
Themes

- Syntax-driven/tree-oriented programming (attribute grammars).
- Principles of programming languages
- Formal semantics
- Type systems
- Lattice theory, fixpoint iteration and monotone functions
- Theory into practice: everything implemented.
What you can expect to get out of this course

▶ Syntax-driven/tree-oriented programming (attribute grammars)
▶ A technical look at typical programming-language constructs.
▶ Static analysis as an approximation of the meaning of a program
▶ The analysis of first-order and higher-order languages
▶ The mathematics in order to understand the technicalities
▶ Implementation of program analysis and transformation
▶ Some more advanced topics (tbd).
Course organisation
Course form

- Lectures: (about) 2 × 2 hours per week.
  - First: focus on lab exercises
  - Later: capita selecta
- And: after each lecture
  - Lab exercises operationalize the theory
  - Organisation: pairs for labs
- Early on in the course more lecture, less lab.
- Assignments:
  - Lab: Static analysis of first-order languages (30%)
  - Lab: Static analysis of higher-order languages (30%)
- Exam: all material of the course (40%)
Prerequisites

- Participants are assumed to be familiar with the basic concepts of imperative and functional programming.
- Advanced functional programming is not a prerequisite.
- During the course, we will implement everything in Haskell.

- Deviation is allowed in special circumstances
- Experience with combinator-based parsing is assumed, but not always necessary.
Course material

- **Slides/handouts, assignments:** made available on the course website
- **Software:** stack, starting templates will install all dependencies via stack
- **Reading material:** a book, some papers
- **Exercises:** in the book and old exams
Further reading: TAPL


Further reading: Dragon book

Further reading: Tiger books


Further reading: Grune et al.

Further reading: Mitchell
