Verification of Functional Programs

Advisor: **Benedikt Maderbacher**

Motivation

Pure functional programs are uniquely suited for formal verification. There exists many techniques that would be impossible to implement for imperative programs. Often the specification of the program is given as a type signature. However, these type systems are much more powerful than e.g. Java's. We can express most specifications as types and it will be automatically verified by the type checker. Some times the type checker will need a little help, to understand your types.

For this project you should get familiar with one functional programming language and its verification capabilities. Some possibilities:

- **Liquid Haskell** An extension of Haskell's type system with refinement types. Types can be restricted with predicates. Uses an SMT solver for type checking. https://ucsd-progsys.github.io/liquidhaskell-blog/
- **Idris** A dependently typed programming language. It can do arbitrary computations on the type level. https://www.idris-lang.org
- **Stainless** A verification tool for Scala. Specifications are provided as annotations for pre- and post-conditions. https://stainless.epfl.ch

Goals and Tasks

- Pick a functional programming language or a verification tool
- Learn how to use it
- Implement and verify a short program

Literature

- N. Vazou, E. L. Seidel, and R. Jhala
  LiquidHaskell: experience with refinement types in the real world
  Proceedings of the 2014 ACM SIGPLAN symposium on Haskell
- E. Brady
  Type-driven development with Idris
  Manning, 2017

Courses & Deliverables

- ☑ **Introduction to Scientific Working**
  Short report on background
  Short presentation
- ☑ **Bachelor Project**
  Project code and documentation
- ☑ **Bachelor's Thesis**
  Project code
  Thesis
  Final presentation

Recommended if you're studying

☑ CS ☑ ICE ☑ SEM

Prerequisites

- Basic knowledge of functional programming
- Interest in logic and type systems

Advisor / Contact

benedikt.maderbacher@iaik.tugraz.at