

Bachel Thesis 2020 **DRAFT**

Ramon Rüttimann

# Functional Go

## Situation

## The programming language Go (Golang) has been gaining popularity since its inception in 2009, mainly as a platform programming language and as a direct competitor to C++ or Java. Though it is often claimed to be extremely fast (which probably comes from its similarity to C), it is still a garbage collected language and thus similar or on-par with the performance of Java. There have been proposed compiler optimisation that have been turned down for reasons like ease of debugging (stack tracing, performance profiling), with the famous example of Tail Call Optimisation. Go also supports functional programming concepts like first class functions. Purely functional languages have an even bigger surface for optimisations, thanks to the lambda calculus.

## Objective

## In this bachelor thesis, compiler optimisations from purely functional languages should be examined and benchmarked. This should be done by writing a baseline benchmark in a functional programming language of choice (e.g. Haskell), an idiomatic, imperative version in Go and a functional version in Go. Then, different compiler optimisations will be targeted, and all different versions benchmarked against each other. This will result in 4 benchmarks (Haskell, idiomatic Go, functional Go before and after compiler optimisations). Which optimisations to apply has to be researched, one obvious target would be Tail Call Optimisation. This comparison should indicate possible optimisations in the Go compiler, the drawbacks of implementing these (maybe also in compile-time), and should demonstrate the performance between specifically optimized languages and language constructs.

## Submission Date

Friday, July 5th, 2020

Winterthur, February 10th, 2020

Gerrit Burkert, Karl Rege