Functional Programming and Applications

Prof. Me. Alexandre Garcia de Oliveira -
Prof. Felipe Cannarozzo Lourenço
alexandre.oliveira@fatec.sp.gov.br -
felipe.lourenco01@fatec.sp.gov.br

30 de julho de 2023

1 Objective

The main goal of this short-term course is to provide students with a way of reasoning about functional programs and also to equip them with the tools and techniques needed to solve computer science problems. Another goal is to introduce students to the Haskell and Agda programming languages, as well as provide an introduction to more advanced topics such as type theory and category theory

2 Syllabus

Introduction to Haskell and cabal. Lists and Tuples. Algebraic data types (ADTs) and pattern matching. Lambdas. Currying. High-order functions. Operations on lists: map, filter, and fold. Parametric Polynorphism. Type-classes. Semigroups and Monoids. Equational Reasoning with Agda and type theory basics (if we have time). Basics of Category Theory. Functors. Applicative Functors. Monads and IO.

3 Methodology

This short-term course will explain each topic in simple terms. Additionally, a computer science challenge will be provided to help students grasp the

functional programming way of thinking.

4 Schedule

- Monday: Intro, Haskell basics, list operations, type, recursion.
- Tuesday: Recursive types, lambdas, currying, high-order functions, map, filter, and fold. Cabal projects.
- Wednesday: Parametric Polymorphism, Typeclasses, Semigroups and Monoids, Intro to Category Theory, Functors, Applicatives, and Monads.
- Thursday: IO monad, parsers, and project related stuff. Type theory and Agda if we have time.
- Friday: Project related stuff and final project presentation.

5 Course Repository

https://github.com/romefeller/haskell-short-term-2023/

6 The Final Project

The final project will be a collaborative effort, completed in pairs. This project will revolve around the creation of DSL (Domain Specific Language) interpreters. Each team is tasked with developing a feasible interpreter mockup to be presented during the final lecture on Friday. The interpreter should have the following components:

- A succinct description detailing its functionality and potential applications.
- A unique name for the DSL, accompanied by an optional logo.
- A GitHub repository dedicated to the project, featuring a README that outlines the primary objectives of the DSL.

- A well-structured Abstract Syntax Tree (AST) and its corresponding interpreter, both of which should be represented within a Haskell module.
- The syntax of the DSL, also contained within a separate Haskell module. Aim for simplicity and narrow the scope when needed.
- A functionality for reading files written in your DSL syntax and producing the end result of any program.
- A test folder populated with programs written in your DSL. These will be instrumental during the presentation.
- A brief overview of potential challenges, pitfalls, and bugs associated with the interpreter.

Teams should actively collaborate on GitHub, ensuring that all commits are accompanied by meaningful messages. Each presentation will be allocated a maximum duration of 15 minutes. Teams have the freedom to structure and deliver their presentations as they wish.

7 Tools

- GHC compiler (The Glorious Glasgow Haskell Compiler) 8.6.5;
- ghcup, cabal, and other Haskell related tools;
- Agda and emacs;

8 Referências

1. OLIVEIRA, A. G. Haskell: Uma introdução à programação funcional. Casa do código. 2017.
2. LIPOVACA, Miran. Learn you a Haskell for a great good, 2015.
3. O'SULLIVAN, Bryan. Real World Haskell. Oreilly, 2008.
4. WADLER. P., Programming foundations in Agda. https://plfa.github.io/