ECE491: Advanced Compilers Topics
Independent Study Proposal

Prof. Sable
Jonathan Lam

2022/01/20

1 Overview

The goal of this course is to explore the student's interests in advanced compilers topics not covered in ECE466: Compilers. The topic will be designing a compiler for a functional programming language, following a tutorial (listed in the references) by the creator of Haskell. Functional languages have been well touted for their many benefits over imperative programming, such as their use in compositional programs (which is a tenet of good software design), their use of immutable programming (another tenet of good design, and commonly used in formal program analysis), advanced type systems (e.g., the Hindley-Milner type inference system), and the high degree of optimization despite their high level of abstraction from the underlying architecture.

The workload will be almost completely self-study, following the tutorial. The tutorial is organized into six chapters: chapter 1 involves the definition of the functional language and data structures; chapters 2-5 involve different compiler implementations; and chapter 6 implements a useful language extension. Some of the necessary code is provided in the tutorial, and some is left as an exercise. After each major implementation (more or less after each chapter) a demo and explanation will be provided to the advisor.

2 Workload and deliverables

There will be weekly readings. The capstone project will be four different implementations for a compiler for the hypothetical language "Core" following the course materials. These compilers will generate low-level representations of the Core language that can be executed on well-defined abstract machines.

3 Resources

- Implementing Functional Languages: a tutorial (by the creator of Haskell and a Turing Award winner) `https://www.microsoft.com/en-us/research/wp-content/uploads/1992/01/student.pdf`
- Write you a Haskell. `http://dev.stephendiehl.com/fun/`

4 Timeline

Each chapter is roughly 40-60 pages and is estimated to take 2-3 weeks for self-study. The timeline below is designed with this in mind. Completing each chapter is a major milestone, and a brief demo and verbal explanation will be given after completing each chapter.

Weeks 1-2 Chapter 1: Introduction to the Core language and project setup

Weeks 2-3 Chapter 2: Template instantiation

Weeks 4-6 Chapter 3: The G-Machine

Weeks 7-9 Chapter 4: TIM: the three instruction machine

Weeks 10-12 Chapter 5: A Parallel G Machine

Weeks 13-15 Chapter 6: Lambda lifting