Assignment 2 Proposal of HasWasm: A WebAssembly-like Embedded DSL in Haskell

Faiz Ilham Muhammad (5231981) and Xinliang Lu (0822760)

02 June 2023

1 Problem Statement

WebAssembly (Wasm) is an assembly-like language that is used in many major browsers with focus on fast execution speed and security-by-sandboxing. While the most common use case of Wasm is for compilation target for other programming languages, there are some cases that need a manually build Wasm code.

However, user-friendly, expressive, and type-safe language abstractions for creating Wasm code are still lacking, especially for Haskell. While there are a few Haskell libraries that can be use for constructing Wasm code such as binaryen binding[1], those libraries can only provide type checking during runtime. To address this problem, we propose to implement a Haskell-based embedded DSL for constructing Wasm code as our project.

2 Approach and Evaluation Questions

2.1 Approach

The goal of the project is to provide an embedded DSL for WebAssembly programming in Haskell that is type-safe and expressive. To accomplish this, we will utilize Haskell's type system, especially its Generalized Algebraic Data Types (GADT) feature.

The project will be divided into the following phases:

1. **Defining the DSL's syntax and semantics.** We will first define the syntax and the semantics of the DSL. Because of the limited time, we will only implement a subset of WebAssembly grammar and features.
2. **Implementing the DSL.** We will then proceed to implement the DSL in Haskell. This includes the API and functions for constructing each WebAssembly instructions and structures, and a AST data type for representing the constructed WebAssembly module.
3. **Testing the DSL.** We will test the DSL through a variety of test cases
4. **Creating a Wasm code generator.** We aim to create a Wasm code generator that converts the AST into an executable Wasm module.
5. **Implement a simple Wasm program optimizer.** Although this is not the main target of the project, we will also aim to create a Wasm program analysis that may do simple optimization before generating the executable, for example dead code elimination.

2.2 Evaluation Questions

1. How effective does the DSL ensure the type correctness of the constructed Wasm code?
2. How does the DSL compare to existing methods of generating Wasm code in terms of expressiveness and usability?

[1] https://hackage.haskell.org/package/binaryen

3 Expected Result and Deliverables

The expected result of this project is an embedded DSL of a subset of WebAssembly in Haskell, which ensures type correctness in compile time and expressive enough to use. Currently we aim to implement the DSL for a subset of the WebAssembly with following features:

1. Two basic types (i32 and f32), with most of the arithmetic and comparison instructions.
2. Blocks and control flows (if-else, loop, jumps)
3. Functions
4. Local and global variables
5. Export and import functions

Key deliverables of the project include:

1. A complete implementation of the DSL for the selected subset of WebAssembly.
2. A comprehensive suite of test cases.
3. Detailed documentation of the DSL, including examples of common use cases.

4 Project Risks

We found that doing type-level programming is quite challenging, so we might not have enough time to implement more advanced features like optimizing the generated Wasm codes. We estimate the following outcomes:

Best outcome: We successfully implement all the features listed in the Approach Section. Our design of the DSL is expressive enough and easy to use.

Worst outcome: We can only achieve the first three items in Approach Section. The DSL may also be too abstract or difficult to use.

5 Related Works

While we do not found a Haskell-based embedded DSL for WebAssembly, there is a Haskell library for building WebAssembly code called binaryen. The library is a Haskell binding to the WebAssembly binaryen toolkit's C API.