A Java Bytecode Generator for the Whitespace Programming Language Implemented in Clojure

James You 001419572,
Zichen Jiang 001320889

February 28, 2018

Abstract

We propose a Clojure-based java bytecode compiler for the esoteric programming language Whitespace. Two artifacts are intended to be generated from the compiler: a user readable intermediate representation and a Java virtual machine `.class` file. We also intend to implement specific program optimizations to reduce redundancy and inefficiency. A report will discuss challenges and details of the project at its conclusion.

1 Areas of Focus

1.1 Intermediate Code Generation

The domain of the Java virtual machine was specifically chosen for its widely accessible code generation tools and platform compatibility. Since Whitespace is a stack-oriented programming language, all operations which exist in Whitespace must also have an equivalent set of stack instructions in the Java virtual machine.

1.2 Program Optimization

Because of the stack-oriented nature of Whitespace, we know it is possible from previous work in compiler optimization that we can apply well developed optimization techniques such as Peephole optimization to either the original Whitespace program or the generated Java bytecode.

1.3 Functional Languages for Compiler Construction

Using a functional language for the implementation of the bytecode compiler will greatly simplify the areas of interests given above. Functional features such as pattern matching can be used for both code generation and optimization.

2 Implementation

It is intended the compiler infrastructure be implemented in Clojure, a functional language with allowance for side effects. The generated class files will make use of the Apache Software Foundation's ByteCode Engineering Library and the intermediate code representation will be provided in Jasmin, which itself is compilable to bytecode as well.

3 Relevance

Whitespace is considered to be an *esoteric* programming language, meant to explore the boundaries of programming languages. We hope to provide an accessible platform generic Whitespace compiler for the programming languages community in spirit of esolangs and if possible, have the project incorporated into a public collection of Whitespacers (Whitespace compilers).