Project Proposal: Grammar Based Source Coding in Compiler Optimization

WU, Chenhao 117010285

February 22, 2019

1 Introduction

In computer science, the compiler optimization is generally processed using a sequence of optimizing transformations, in which will produce an equivalent output program that uses fewer computing resources by algorithms closely related to the language structures and behaviors. In the early years of computer science, optimizations of compilers are being performed basically by detecting and resolving specific cases which might cause unnecessary payload in CPU or memories. The shortages of this approach are obvious that optimization are highly programming language-dependent such that for example, the optimization for C++ compiler will not be able to be migrated for Swift compiler since that they have different cases need to be optimized due to different language structures. Also, for a determined program language the optimizations are not that straightforward to be enumerated because the behavior of machine, third-party library and even programmers are sometime unpredictable and undecidable.

In recent years computer scientists have made attempts to apply statistical approaches to optimize the compilers, such as profile-guided optimization (PGO) and Milepost GCC, which

will learn the *language grammar* (statistical models) of source code and implicitly remark different execute-priority according to the ranking of the probability distribution of scopes.

Inspired by the statistical approaches of compiler optimization, an primitive thought is to regard the programming languages as arbitrary languages with its grammar, and regard compiler optimization as an process to compress the source code using grammar based source coding in order to reduce the duplicated operations.

2 Motivation

The objective of profile-guided optimization and Milepost GCC mentioned above is to optimize operations which might not be enumerated before the source coding being given, that for example, for a long *if-else* scope, PGO can pre-calculate the probability of each *if* branch and reform the order of *if-else* scope according to the probability distribution.

These approaches can then be abstracted into a context-free grammar based source coding with a variable set V consists of every possible scope in the source code and a language syntax P. However, due to the significant growth of Artificial Intelligence in recent years, tons of open-source third party libraries can be found and downloaded online, the compiler optimization based on a context-free grammar seems not capable in handling unknown packages which are imported in the source code. Hence, we want to research on an universal lossless source code with specification to compiler optimization, in order to learn the unknown language structures from third party packages or even from different programming languages, and then apply optimization on the compiling.

Also, by this approach we want to show some insight of programming language itself, which might also be associated with the human logic and reasoning.