Eötvös Loránd Tudományegyetem
Informatikai Kar
Programozási Nyelvek és
Fordítóprogramok Tanszék

Applying slicing algorithms on large code bases

Tibor Brunner
doktorandusz

Olivér Hechtl
programtervező informatikus MSc

Budapest, 2017

Contents

Chapter 1

Introduction

"Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?" - Brian Kernighan

Nowadays, there are a lot of tools available for debugging. A developer can examine call stacks, variable informations, and so on. There are a lot of times when a bug is when some variable does not behave like the writer of that code part would expect. When the programmer discovers it, he must follow back the path of assignments, and find out where did it get an unexpected value. Program slicing targets this kind of debugging. We can define a program slice as a part of the program which contains a selected statement, and all other statements which influences it, or gets influenced by it. These two types are respectively called backward and forward slicing. Basically this is what many programmers do intuitively, when facing with bugs. In this thesis, I'll introduce a few approaches for computing slices, and describe a working prototype tool application, which can analyze C++ programs, and compute slices of them.

Something about maintenance time vs coding time. Something about slicing in general. Something about large codebases. Something about Clang/LLVM compiler infrastructure and it's usefulness in applying slicing to c++.

Chapter 2

slicing, methods and efficiency

Chapter 3

LLVM/Clang infrastructure

Chapter 4

Implementation and algorithm

Bibliography