CS4215 Project Report

Team Estepona - Abstract Abstract Machines

Tay Jing Xuan Tee Weile Wayne

April 2022

1 Abstract

We implemented an abstract interpreter for a subset of source. By approximating program states, our abstract interpreter always terminates and can be used to analyse program behaviour and allow for code optimisation. Our interpreter also allows for visualisation of the abstract state-space for analysis.

2 Introduction

The objective of static program analysis is to determine the behaviour of a program without running it. This includes determining whether the program terminates, whether optimisations can be applied, and what states are reachable during program execution.

This is a difficult problem. The halting problem for example is undecidable. Data-flow analysis is also difficult to perform on languages such as Source which support higher-order functions.

A key difficulty is that a virtual machine executing a program might require an infinite number of states for a non-terminating program. However, by following the abstracting abstract machines approach [1], we are able to convert a concrete virtual machine into a non-deterministic abstract machine with finite states, with each abstract state representing multiple possible

concrete states. As a result, we are able to soundly approximate program behaviour in finite time.

For example, if the concrete machine transitions from state A to state B, then the abstract machine must transition from some abstract state A' to some abstract state B', where A' and B' are the abstract states representing A and B. If the abstract machine always terminates, then so must the concrete machine.

We applied this approach to convert the Source virtual machine into an abstract interpreter which supports higher-order functions and conditionals. The interpreter provably terminates, returning a graph of abstract states reached that can be visualised by the user and potentially used to derive complier optimisations.

References

[1] David Van Horn and Matthew Might. Abstracting abstract machines. In *Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*, pages 51–62, 2010.