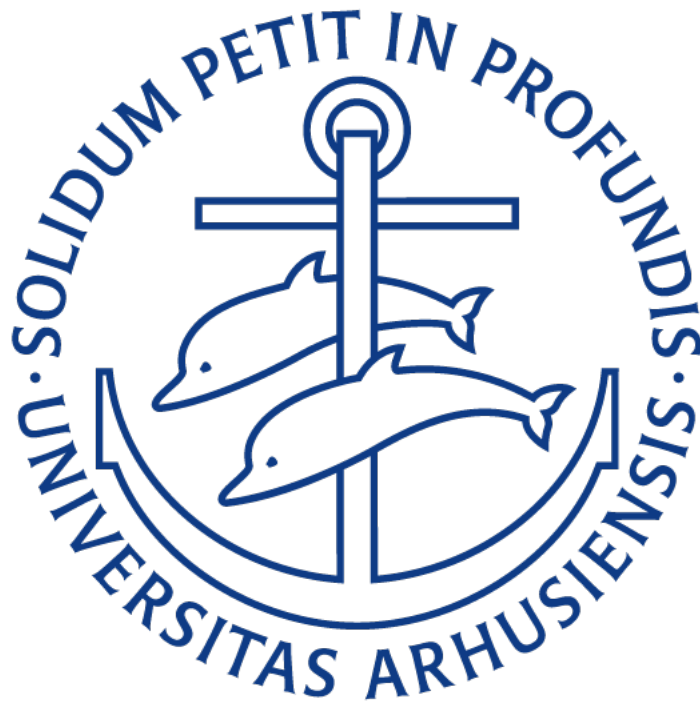


Symbolic Execution(Working title)

Aarhus Universitet



Søren Baadsgaard

February 2, 2019

Abstract

Contents

1	Introduction	2
2	Summary of theory	3
3	Basic symbolic execution for the <i>SImPL</i> language	4
3.1	description	4
3.2	Introducing the <i>SImPL</i> language	4
4	Further extensions	6
5	Conclusion	7
A	Source code	8
B	Figures	9

Chapter 1

Introduction

Chapter 2

Summary of theory

Chapter 3

Basic symbolic execution for the *SImPL* language

3.1 description

In this chapter we will describe the process of implementing symbolic execution for a simple imperative language called *SImPL*.

3.2 Introducing the *SImPL* language

SImPL (**S**imple **I**mpерative **P**rogramming **L**anguage) is a small imperative programming language, designed to highlight the interesting use cases of symbolic execution. The language supports only one type, namely the set integers \mathbb{N} , where 0 and 1 doubles as the logic values *true* and *false* respectively. *SImPL* supports basic variables that can be assigned the value of any expression, as well as basic branching functionality through an **If - Then - Else** statement. Furthermore it allows for looping through a **While - Do** statement.

We will describe the language formally, by the following Context Free Grammar:

$$\langle int \rangle ::= 0 \mid 1 \mid -1 \mid 2 \mid -2 \mid \dots$$

$$\langle var \rangle ::= a \mid b \mid c \mid \dots$$

$$\begin{aligned} \langle exp \rangle ::= & \langle int \rangle \\ & \mid \langle var \rangle \\ & \mid \langle exp \rangle + \langle exp \rangle \mid \langle exp \rangle - \langle exp \rangle \mid \langle exp \rangle * \langle exp \rangle \mid \langle exp \rangle / \langle exp \rangle \\ & \mid \langle exp \rangle > \langle exp \rangle \mid \langle exp \rangle == \langle exp \rangle \\ & \mid (\langle exp \rangle) \end{aligned}$$

$$\begin{aligned} \langle stm \rangle ::= & \langle var \rangle = \langle exp \rangle \\ & \mid \langle stm \rangle \langle stm \rangle \\ & \mid \text{if } \langle exp \rangle \text{ then } \langle exp \rangle \text{ else } \langle exp \rangle \\ & \mid \text{while } \langle exp \rangle \text{ do } \langle exp \rangle \\ & \mid \end{aligned}$$

where $+, *, -, /$ denotes the usual arithmetic operators on integers, and $>, ==$ denotes the comparison-operators of *greater-than* and *equal-to* respectively.

Chapter 4

Further extensions

Chapter 5

Conclusion

Appendix A

Source code

Appendix B

Figures