Thomas Fahringer - Advanced Symbolic Analysis for Compilers - 2003

Tom Rochette <tom.rochette@coreteks.org>

July 24, 2025 — daae079c

0.1 Context

0.2 Learned in this study

0.3 Things to explore

- Is static analysis memory consumption a good representation of how much resources is required by a programmer to understand a piece of code?

1 Overview

2 Notes

2.1 1. Preface

- The objective of program analysis is to automatically determine the properties of a program
- Advanced program analysis can:
  - help to find program errors
  - detect and tune performance-critical code regions
  - ensure assumed constraints on data are not violated
  - tailor a generic program to suit a specific application
  - reverse-engineer software modules
  - etc.
- Many symbolic analyses are based on abstract interpretation and an abstraction of program input data
- The program context:
  - variables values
  - assumptions about and constraints between variable values
  - conditions under which control flow reaches a program statement

2.2 2. Introduction

```
read(A, B)
A := A + B
B := A - B
A := A - B
```

- Instead of computing numbers for variable A and B we assign them symbolic values. Let us assume that variable A and B are undefined. We denote this undefined variable binding of A and B as follows
  - $A = \perp$
  - $B = \perp$

- As A and B are assigned, we assign them a symbolic value where ∇_1 and ∇_2 denote the input values of the read statement
  - $A = \nabla_1$
  - $B = \nabla_2$
- We propagate symbolic value of A and B (after line 2)
  - $A = \nabla_1 + \nabla_2$
  - $B = \nabla_2$
- At line 4
  - $A = (\nabla_1 + \nabla_2) - \nabla_1$
  - $B = (\nabla_1 + \nabla_2) - ((\nabla_1 + \nabla_2) - \nabla_1)$

2.3 2.3 Constributions

2.4 2.3.1 Symbolic Analysis Framework

- Program contexts include three components: variable values, assumptions about and constraints between variables values, and path condition

2.5 3 Symbolic Analysis of Programs

2.6 3.1 Introduction

- A program context c is defined by a triple [s, t, p] that includes a state s, a state condition t and a path condition p
- State S: Described by a set of variable/value pairs $v_i = e_i$ where v_i is a program (scalar or array) variable and e_i is a symbolic expression describing the value of v_i
- State condition t: Assumptions about variable values are described by a state condition t. Additional constraints on variable values such as those implied by loops (recurrences), variable declaration and user assertions (specifying relationships between variable values) are also added to the state condition
- Path condition p: Describes the condition under which control flow reaches a given program statement

3 See also

4 References

- Fahringer, Thomas, and Bernhard Scholz. Advanced symbolic analysis for compilers: new techniques and algorithms for symbolic program analysis and optimization. Vol. 2628. Springer, 2003.