Notes on The Formal Semantics of Programming Languages

Ray Li

October 16, 2024

Contents

1 Chapter 1: Basic Concepts

Question 1. *What is the difference between operational and denotational semantics?*

Answer 1. *Operational semantics provides a step-by-step procedure to evaluate a program, effectively describing how the program executes. Denotational semantics, on the other hand, maps each program to a mathematical object, often focusing on what the program computes rather than how.*

Comment 1. *This is an important distinction because operational semantics is more intuitive but may lack the mathematical rigor found in denotational semantics.*

Question 2. *Explain the use of abstract syntax in defining programming languages.*

Answer 2. *Abstract syntax ignores details like parentheses and focuses on the structure of the expressions. For example, in arithmetic, $e \to e + e$ is a rule that defines an expression without caring about the concrete appearance of parentheses.*

Comment 2. *I think abstract syntax trees (ASTs) provide an easy way to implement compilers by focusing on structure instead of syntax.*

[Ray's Note 1: I need to explore more about the connection between operational semantics and compiler design.]

2 Chapter 2: Inductive Definitions

Question 3. *How can inductive definitions be used to define programming language semantics?*

Answer 3. *Inductive definitions allow us to define objects step by step. In programming languages, they are used to define the structure of programs (syntax) and the way programs are evaluated (semantics). An example is defining valid expressions in a language using base cases and recursive steps.*

Comment 3. *Inductive definitions seem to be particularly useful when defining operational semantics.*

3 Ray’s Notes Ssummary

Ray’s Note 1: