Language Engineering Notes

paraphrased by Tyler Wright

An important note, these notes are absolutely ***NOT*** *guaranteed to be correct, representative of the course, or rigorous. Any result of this is not the author's fault.*

1 Syntax and Semantics

1.1 What is Syntax?

Syntax is the grammatical structure of a program. For example, for the program $x := y; y := z; z := x;$, syntactic analysis of this program would conclude that we have three statements concluded with ';'. Each of said statements are variables followed by the composite symbol ':=' and another variable.

1.2 What are Semantics?

The semantics of a program are what the program evaluates to or rather, the meaning of a syntactically correct program. For example, $x := y;$ evaluates to setting the value of x to the value of y.

2 Operational Semantics

2.1 Overview of Operational Semantics

An operational explanation of the meaning of a construct will explain how to execute said construct. For example, in C, the semicolons provide chronology and the = symbol demonstrates assignment. These statements are abstractions as they do not concern themselves with the specific memory addresses or registers. Thus, these semantics are independent of machine architecture.

2.2 Derivation Trees and Natural Semantics

A program's execution can be modelled by a 'derivation tree' where the higher parts of the tree are breakdowns of the statements below. For example, the program $x := y; y := z; z := x;$ can be written as:

$$\frac{}{\langle x := y; y := z; z := x;, s_0\rangle \to s_3}$$