Chapter 1: Introduction

1. Programming Languages Classification

a. declarative languages (Implicit)

- functional – lisp, Scheme, ML, Haskell

- logical – Prolog

- dataflow – Simulink , Scala

b. imperative languages (Explicit)

- procedural

- object oriented

2. What makes a successful programming language?

3. Compilation and Interpretation

Compilation – compiles into an executable program written in machine language.

Interpretation – directly interpreted and the execution is simulated by the interpreter.

a. What is the main difference between compilation and interpretation?

+ compilation and execution on virtual machines

+ preprocessing: what is its relation with compilation?

+ IDE definition

- Compiler phases and what each phase does

Chapter 2: Syntax

- lexical analysis

+ tokens and regular expressions

- how to write regular expressions for different types of tokens

+ regular expressions and NFA

- algorithm to convert a regular expression to the corresponding NFA.

- NFA to DFA

- syntax analysis

+ derivation and the language recognized by a grammar

+ context free grammars

- writing context free grammar for a language

+ leftmost and rightmost derivation

+ parse tree

+ ambiguous grammars

+ Parsing

- two general approaches:

+ Recursive descent parsing

+ LL(1) Predictive parsing

- step by step trace of the LL(1) parsing algorithm

- construct LL(1) parsing table

+ algorithm for computing First and Follow sets

+ Using First and Follow sets to compute the parsing table

Chapter 4: Semantic Analysis

- static semantics and dynamic semantics

- attribute grammars

+ writing an attribute grammar

- adding semantic rules on a given grammar to perform some function

- writing an attribute grammar to enhance the underlying context

free grammar.

- building decorated parse tree (parse tree annotated with attributes)

- synthesized and inherited attributes

- S-attributed and L-attributed grammars