Chapter 3 – Describing Syntax and Semantics

3.1 Introduction

* Syntax- the form of a programming language’s expressions, statements, and program units
* Semantics- the meaning of those expressions, statements, and program units
* Example: Java while statement
  + Syntax:  
    while (boolean\_expr) statement
  + Semantics: when the current value of the Boolean expressions is true, the embedded statement is executed. Then control implicitly returns to the Boolean expression to repeat the process. If the Boolean expression is false, control transfers to the statement following the while construct

3.2 The Genera Problem of Describing Syntax

* Sentences- the strings of a language, also can be called statements
* Lexemes- small units of formal descriptions of the syntax of programming languages that often do not include descriptions of the lowest-level syntactic units
  + Description can be given by lexical specification, which is usually separate from the syntactic description of the language
* Token- a category of a language’s lexemes
  + Examples:
    - Index (lexeme), identifier (token)
    - = (lexeme), equal\_sign (token)
    - + (lexeme), plus\_up (token)
* Language Recognizers- languages can be defined by recognition and by generation
* Language Generators- a device that can be used to generate the sentences of a language

3.3 Formal Methods of Describing Syntax

* Backus-Naur Form of and Context-Free Grammars
  + Context-Free Grammars
  + Backus-Naur Form- revised method of syntax description
* Fundamentals
  + Metalanguage- language that is used to describe another language
    - Example: BNF is a metalanguage for programming languages  
      <assign> 🡪 <var> = <expression>
  + Left-hand side (LHS)- the abstraction being defined
  + Right-hand side (RHS)- the definition of the LHS
  + Rule- the definition of the LHS and the RHS
  + Nonterminal symbols- abstractions in a BNF description, or grammar
  + Terminal symbols- lexemes and tokens of the rules
  + Grammar- a collection of rules
* Describing Lists- BNF uses recursion
* A rule is recursive if the LHS appears in its RHS
* Grammars and Derivations
  + Start symbol- a special nonterminal of the grammar that start a sequence of applications of the rules
  + Derivation- sequence of rule applications
  + Sentential form- each of the strings in the derivation
  + Leftmost derivations- derivations that use the sentential form order of replacement
* Parse Trees- hierarchical structures that describe the sentences of the language they define
* Ambiguity- a grammar that generates a sentential form for which there are two or more distinct parse trees
* Operator Precedence
* Associativity of Operators
  + Left Recursive- When a grammar rule has its LHS also appearing at the beginning of its RHS
  + Right Recursive- If the LHS appears at the right end of the RHS
* An Unambiguous Grammar for if-else
* Extended BNF
  + Meta symbols- brackets, braces, and parenthesis that are notational tools and not terminal symbols in the syntactic entities they help describe
* Grammars and Recognizers