Syntax – The form or structure of the expressions, statements, and program units

Semantics – The meaning of the expressions, statements, and program units

Syntax and Semantics provide a language’s definition

* Users of a language definition
  + Other language designers
  + Implementers
  + Programmers (the users)

The General Problem of Describing Syntax: Terminology

* A *lexeme* is the lowest level syntactic unit of a language (\*, sum, begin)
* A *token* is a category of lexemes (identifier)
* A *sentence* is a string of characters over some alphabet
* A *language* is a set of sentences

Formal Definition of Languages

* Recognizers
  + A recognition device reads input strings over the alphabet of the language and decides whether the input strings belong to the language
  + Example: Syntax analysis part of a compiler
    - Detailed discussion appears in Chapter 4
* Generators
  + A device that generates sentences of a language
  + One can determine if the syntax of a particular sentence is syntactically correct

BNF and Context-Free Grammars

* Context-Free Grammars
  + Developed by Noam Chomsky in 1950s
  + Language generators, meant to describe the syntax of natural languages
  + Define a class of languages called context-free languages
* Backus-Naur Form (1959)
  + Invented by John Backus to describe the syntax of ALGOL 58
  + Is the equivalent of Context-Free

BNF Fundamentals

* Abstractions are used to represent classes of syntactic structures, they act like syntactic variables (also called nonterminal symbols, or just nonterminals
* Terminals are lexemes or tokens
* A rule has a left-hand side (LHS), which is a nonterminal, and a right-hand side (RHS), which is a string of terminals and/or nonterminals
* Nonterminals are often enclosed in angle brackets
  + <ident\_list> -> identifier
* Grammar: a finite non-empty set of rules
* A start symbol is a special element of the nonterminals of a grammar

Describing Lists

* Syntactic lists are described using recursion
  + <ident\_list> -> ident
    - * | ident, <ident\_list>
* A derivation is a repeated application of rules, starting with the start symbol and ending with a sentence (all terminal symbols)

Grammar

1. <sentence> - <noun\_phrase><verb\_phrase>
2. <noun\_phrase> - <article><noun>
3. <article> - a, the
4. <noun> - girl, dog
5. <verb\_phrase> - <verb><noun\_phrase>
6. <verb> - sees, pets

* Q: How to produce “the girl sees a dog”
  + <sentence>
  + => <noun\_phrase><verb\_phrase>
  + => <article><noun><verb\_phrase>
  + => the <noun><verb\_phrase>
  + => the girl <verb\_phrase>
  + => the girl <verb><noun\_phrase>
  + => the girl sees <noun\_phrase>
  + => the girl sees <article><noun>
  + => the girl sees the <noun>
  + => the girl sees the dog
* An Example Programming Grammar
  + <program> -> <stmts>
  + <stmts> -> <stmt>, <stmt>; <stmts>
  + <stmt> -> <var> = <expr>
  + <var> -> a, b, c, d
  + <expr> -> <term> + <term>, <term> - <term>
  + <term> -> <var>, const
* Example Derivation
  + <program>
  + => <stmts>
  + => <stmt>
  + => <var> = <expr>
  + => a = <expr>
  + => a = <term> + <term>
  + => a = <var> + <term>
  + => a = x + <term>
  + => a = x + const
  + => a = x + 3

Derivations

* Every string of symbols in a derivation is a sentential form
* A sentence is a sentential form that has only terminal symbols
* A leftmost derivation is one in which the leftmost nonterminal in each sentential

Parse Tree - A hierarchical representation of a derivation