Syntax

Chapter 2 of the book

Syntax?

- A language is defined by a syntax and a semantics.
- Syntax: definition of what constitutes a grammatically valid program in that language.
- Semantics: Meaning of the program.
- Reference manuals: describe the syntax and semantics of languages in English.
- Formal definition: Precise description of the syntax and semantics of a language. It is aimed at specialists.
  - Formal - based on mathematics.
- How is the syntax described?
  - Using a language - metalanguage
  - Example: Backus-Naur form (BNF)

Expressions

- Starting point for a language.
- Example: \((-b + \sqrt{b^2 - 4.0 \times a \times c})/(2.0 \times a)\)
- Different notations for an expression.
  - Prefix notation (Example: \(+ 1 2\))
  - Postfix notation (Example: \(1 2 +\))
  - Infix notation (Example: \(1 + 2\))
- Mixfix notation: Operations that do not fit into the prefix, infix and postfix classifications. (Example: if-then-else)
- Examples:
  - Prefix:
    \(* + 20 30 60 =\)
    \(* 20 + 30 60 =\)
  - Postfix:
    \(20 30 + 60 * =\)
    \(20 30 60 + * =\)
- How to evaluate a prefix or postfix expression?

Infix notation

- How do you compute \(5 * 7 + 3 - 1\)?
- Is it \((5 * 7) + (3 - 1)\)?
- Is it \(5 * (7 + 3) - 1\)?
- To deal with the ambiguity: Use parentheses or use a precedence on operators.
  - Example: \(\{*, /\} > \{+, -\}\).
  - * and / have the same precedence. + and − have the same precedence.
  - The expression is: \((5 * 7) + 3 - 1\).
- An operator is said to be left-associative if subexpressions containing the same operator or operator with the same precedence are grouped from left to right to be decoded.
  - Examples:
    - − is left-associative. \(4 - 2 - 1\) is \((4 - 2) - 1\) and not \(4 - (2 - 1)\).
    - \(-5 * 7 + 3 - 1\) is \(((5 * 7) + 3) - 1\).
    - +, * and / are also left-associative.
An operator is said to be right-associative if subexpressions containing the same operator or operator with the same precedence are grouped from right to left to be decoded.

**Examples:**
- Exponentiation is right-associative. \(2^3\) is \(2^{(2^3)}\).
- The assignment symbol is right-associative.

Rooted trees can be used to represent arithmetic formulas.

The root of the tree represents the final value of the computation.

The leaves represent the operands, and the intermediate nodes the operators.

This tree represents the equation \((2 + 5) * (3 + 4) + (1 * 6)\) using conventional arithmetic notation, or \(\{2 5 + 3 4 * 1 6 * +\}\) using reverse Polish notation (as on an HP calculator) (postfix notation).

These formulas can be constructed by appropriately traversing the expression tree, i.e. visiting the nodes in the correct order.

There are three natural orders to visit the nodes of the tree, each of which walks up and down the tree in a recursive manner:

- **In-order** – visit all the left subtree before visiting the root node, then visit the right subtree.

  \[ \text{Infix notation} : 2 + 5 * 3 + 4 + 1 * 6 \]

- **Post-order** – visit the left subtree and right subtrees completely before visiting the root node.

  \[ \text{Postfix notation} : 25 + 34 * 16 * + \]

- **Pre-order** – visit the root node before visiting the left subtree and right subtrees.

  \[ \text{Prefix notation} : + * + 25 * 34 * 16 \]

  A fourth natural order, breadth-first traversal, traverses level-by-level, \(\{+ * + * 1 6 2 5 3 4\}\).

  This involves jumping around from one subtree to another, and hence is not good for evaluating expressions, although breadth-first traversal does have numerous applications in computer science.
if-the-else Tree

- How to represent \( if \ a \ > \ b \ then \ a \ else \ b \) as a tree?

Examples of grammars

- Grammar for a digit:
  
  \[
  \text{digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
  \]
  
  digit is a nonterminal.
  
  0, ..., 9 are terminals.

- Grammar for signed integers:
  
  \[
  \text{integer} \rightarrow \text{digit} \mid \text{digit integer}
  \]
  
  \[
  \text{signedinteger} \rightarrow \text{sign integer}
  \]
  
  sign \( \rightarrow + \mid - \mid \sim \)

- Subset of Java:
  
  \[
  \text{assignmentstatement} \rightarrow \text{variable} = \text{expression}
  \]
  
  expression \( \rightarrow \text{variable} \mid \text{variable} + \text{variable} \)
  
  variable \( \rightarrow x \mid y \mid z \)

- Subset of the English language!
  
  sentence \( \rightarrow \text{noun} \ \text{verb} \)
  
  noun \( \rightarrow \text{bees} \mid \text{dogs} \)
  
  verb \( \rightarrow \text{buzz} \mid \text{bite} \)

- The most common notation for grammars is the BNF (Backus-Naur form) notation.

- A BNF grammar is a set of rewriting rules defined on a set of nonterminal symbols, a set of terminal symbols and a "start symbol".

- Terminals form the basic alphabet from which programs are constructed.
  
  Example: \( \text{int, Object, <} \) are terminals in the BNF description of JAVA.

- Nonterminals identify grammatical categories.
  
  The start symbol identifies the principal grammatical category being defined by the grammar.
  
  Example: programme

- The rewriting rules are written using the following meta-symbols: \( \rightarrow \) and \( \mid \).
  
  - Meta-symbols are symbols of the meta-language.
  
  - \( \mid \) is a separator for alternative definitions (it means 'or').
  
  - \( \rightarrow \) is used to define a rewriting rule (it means 'is')

- A rewriting rule is of the form:
  
  \[
  \text{left hand side} \rightarrow \text{"definition"}
  \]
  
  - The left-hand-side is the name of a grammatical category. It is a nonterminal.
  
  - The right-hand-side is a definition that specifies the grammatical structure of the symbol appearing on the left-hand-side of the rule.
  
  - The right-hand-side of a rule can be any sequence of terminals and nonterminals separated by \( \mid \).

- Designing a grammar is not easy.
  
  - It can lead to problems such as ambiguities in grammars.