Principles of Programming Languages

COMP251: Syntax and Grammars

Prof. Dekai Wu

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong, China

Fall 2006

Part I

Language Description

Language Description: Motivation

“Able was I ere I saw Elba.” — about Napoléon

How do you know that this is English, and not French or Chinese?

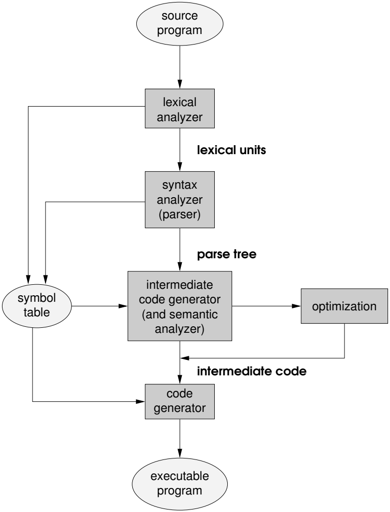
Language Description

A language has 2 parts:

1. **Syntax**
   - lexical syntax
     - describes how a sequence of *symbols* makes up *tokens (lexicon)* of the language
     - checked by a *lexical analyzer*
   - grammar
     - describes how a sequence of *tokens* makes up a valid *program*.
     - checked by a *parser*
2. **Semantics**

   specifies the *meaning* of a program

Compilation



Example 1: English Language

A word = some combination of the 26 letters, a,b,c, ...,z.

One form of a sentence = Subject + Verb + Object.

e.g. The student wrote a great program.

Example 2: Date Format

A date like 06/04/2010 may be written in the general format:

D D / D D / D D D D

where D = 0,1,2,3,4,5,6,7,8,9

But, does 03/09/1998 mean Sept 3rd, or March 9th?

Example 3: Real Numbers (Simplified)

Examples of reals: 0.45   12.3   .98
Examples of non-reals: 2+4i   1a2b   8 <

Informal rules:

- In general, a real number has three parts:
  - an integer part (I)
  - a dot "." symbol (.)
  - a fraction part (F)
- valid forms: $I.F$, $.F$
- I and F are strings of digits
- I may be empty but F cannot
- a digit is one of { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

Expression: Examples

$$a + b \qquad 3 * a + b/c$$

$$\frac{-b+\sqrt{b^2-4*a*c}}{2*a} \qquad \frac{a*(1-R^n)}{1-R}$$

```
if (x > 10) then
     x /= 10
else
     x *= 2
```

c.f. "While I was coming to school, I saw a car accident."
The sentence is in the form of: "While E_1, E_2."

Expression Notation: Example 4

Goal: Add a to b.

Abstract Syntax Tree

Infix : $a + b$

Prefix : $+ab$

Postfix : $ab+$

```
  +
 / \
a   b
```

Abstract syntax tree is *independent* of notation.

Expression

- A constant or variable is an expression.
- In general, an expression has the form of a function:

$$E \triangleq \mathbf{Op}\,(E_1, E_2, ..., E_k)$$

  where **Op** is the operator, and $E_1, E_2, ..., E_k$ are the operands.
- An operator with k operands is said to have an arity of k; and **Op** is an k-ary operator.

unary operator :	$-x$
binary operator :	$x + y$
ternary operator :	$(x > y)\ ?\ x \ : \ y$

Infix, Prefix, Postfix, Mixfix

- Infix : E_1 **Op** E_2 (must be binary operator!)

$$a + b,\ a * b,\ a - b,\ a/b,\ a == b,\ a < b.$$

- Prefix : **Op** E_1 E_2 $\ldots E_k$

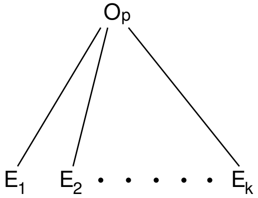
$$+ab,\ *ab,\ -ab,\ /ab,\ == ab,\ < ab.$$

- Postfix : E_1 E_2 $\ldots E_k$ **Op**

$$ab+,\ ab*,\ ab-,\ ab/,\ ab ==,\ ab < .$$

- Mixfix : e.g. **if** E_1 **then** E_2 **else** E_3

Abstract Syntax Tree



Expression Notation: Example 5

abstract syntax tree

infix : $3 * a + b/c$

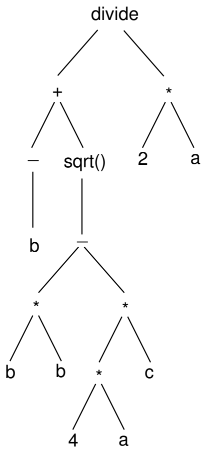
prefix : $+ * 3a/bc$

postfix : $3a * bc/+$

```
        +
       / \
      /   \
     *     /
    / \   / \
   3   a b   c
```

Note: Prefix and postfix notation does <u>not</u> require parentheses.

Expression Notation: Example 6



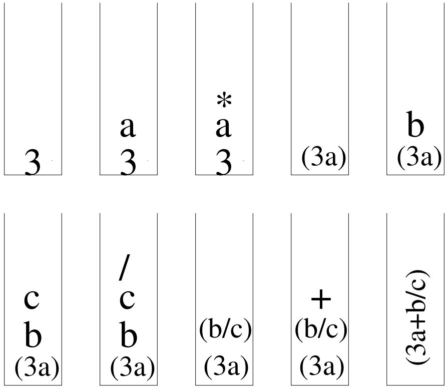
$$\text{infix}: \quad (-b + \sqrt{b^2 - 4 * a * c})/(2 * a)$$

$$\text{prefix}: \quad / + -b \sqrt{\ } - *bb * *4ac * 2a$$

$$\text{postfix}: \quad b - bb * 4a * c * - \sqrt{\ } + 2a * /$$

Postfix Evaluation: By a Stack

- infix expression: $3 * a + b/c$.
- postfix expression: $3a * bc/+$.



Precedence and Associativity in C++

Operator	Description	Associativity
[ ] . $\rightarrow$	array element structure member pointer	LEFT
- ++ - - *	minus increment decrement indirection	RIGHT
* / %	multiply divide mod	LEFT
+ -	add subtract	LEFT
==	logical equal	LEFT
=	assignment	RIGHT

Precedence

Example: $1/2 + 3 * 4 = (1/2) + (3 * 4)$
because $*$, $/$ has a *higher precedence* over $+$, $-$.

Precedence rules decide which operators run first. In general,

$$x \; P \; y \; Q \; z \quad = \quad x \; P \; (\; y \; Q \; z \;)$$

if operator Q is at a higher precedence level than operator P.

Associativity: Binary Operators

Example: $1 - 2 + 3 - 4 = ((1 - 2) + 3) - 4$
because $+$, $-$ are *left associative*.

Associativity decides the grouping of operands with operators of the *same* level of precedence.
In general, if binary operator P, Q are of the same precedence level:

$$x \; P \; y \; Q \; z \quad = \quad x \; P \; (\; y \; Q \; z \;)$$

if operator P, Q are both right associative;

$$x \; P \; y \; Q \; z \quad = \quad (\; x \; P \quad y \;) \; Q \; z$$

if operator P, Q are both left associative.
Question : What if $+$ is left while $-$ is right associative?

Associativity: Unary Operators

- Example in C++: $*a++ = *(a++)$
  because all unary operators in C++ are right-associative.
- In Pascal, all operators including unary operators are left-associative.
- In general, unary operators in many languages may be considered as non-associative as it is not important to assign an associativity for them, and their usage and semantics will decide their order of computation.

Question : Which of infix/prefix/postfix notation needs precedence or associative rules?

Summary on Syntax

- √ Will describe a language by a formal syntax and an informal semantics
- √ Syntax = lexical syntax + grammar
- √ Expression notation: infix, prefix, postfix, mixfix
- √ Abstract syntax tree: independent of notation
- √ Precedence and associativity of operators decide the order of applying the operators