Formal Languages And Compilers
Theory

Christian Rossi

Academic Year 2023-2024

Abstract

The lectures are about those topics:

- Definition of language, theory of formal languages, language operations, regular expressions, regular languages, finite deterministic and non-deterministic automata, BMC and Berry-Sethi algorithms, properties of the families of regular languages, nested lists and regular languages.
- Context-free grammars, context-free languages, syntax trees, grammar ambiguity, grammars of regular languages, properties of the families of context-free languages, main syntactic structures and limitations of the context-free languages.
- Analysis and recognition (parsing) of phrases, parsing algorithms and automata, pushdown automata, deterministic languages, bottom-up and recursive top-down syntactic analysis, complexity of recognition.
- Syntax-driven translation, direct and inverse translation, syntactic translation schemata, transducer automata, and syntactic analysis and translation. Definition of semantics and semantic properties. Static flow analysis of programs. Semantic translation driven by syntax, semantic functions and attribute grammars, one-pass and multiple-pass computation of the attributes.

The laboratory sessions are about those topics:

- Modelisation of the lexicon and the syntax of a simple programming language (C-like).
- Design of a compiler for translation into an intemediate executable machine language (for a register-based processor).
- Use of the automated programming tools Flex and Bison for the construction of syntax-driven lexical and syntactic analysers and translators.

Contents

Chapter 1

Syntax

1.1 Formal language theory

A *formal language* consists of words whose letters are taken from an alphabet and are well-formed according to a specific set of rules.

Definition

An *alphabet* is a finite set of elements called terminal symbols or *characters*. The *cardinality* of an alphabet

$$\Sigma = \{a_1, a_2, \dots, a_k\}$$

is the number of characters that it contains: $|\Sigma| = k$. A *string* or word is a sequence of characters.

Example : The alphabet $\Sigma = \{a, b\}$ has a cardinality of two. Some possible languages derived from this alphabet can be:

- $L_1 = \{aa, aaa\}$
- $L_2 = \{aba, aab\}$
- $L_3 = \{ab, ba, aabb, abab, \dots, aaabbb, \dots\}$

Definition

Given a language, a string belonging to it is called a *sentence* or *phrase*. The *cardinality* or size of a language is the number of sentence it contains. If the cardinality is finite, the language is called *vocabulary*.

Example : Given the language (that is a vocabulary) $L_2 = \{bc, bbc\}$ we have that its cardinality is equal to two.

Definition

The number of repetitions of a certain letter in a word is called *number of occurrences*. The *length* of a string is the number of its elements. Two strings are *equal* if and only if:

- They have the same length.
- Their elements, from left to right, coincide.

Example : The number of occurrences of a and c in aab is indicated with:

$$|aab|_a = 2$$

$$|aab|_c = 0$$

The length of the string aab is equal to:

$$|aab| = 3$$

In order to manipulate strings, it is convenient to introduce several operations:

- Concatenation: given two strings $x = a_1a_2 \dots a_h$ and $y = b_1b_2 \dots b_k$ this operation is defined as:

  $$x \cdot y = a_1a_2 \dots a_hb_1b_2 \dots b_k$$

  Concatenation is non-commutative and associative ($x(yz) = (xy)z$). The length of the result is the sum of the length of the concatenated strings ($|xy| = |x| + |y|$).

- Empty string: ε is the neutral element for concatenation that satisfies the identity:

  $$x\varepsilon = \varepsilon x = x$$

  It is important to note that $|\varepsilon| = 0$ and that the set that contains this operator is not the empty set.

- Substring: let string $x = xyv$ be written as the concatenation of three, possibly empty, strings x, y and v. Then, strings x, y and v are substrings of x. Moreover, string u is a prefix of x and v is a suffix of x. A non-empty substring is called proper if it does not coincide with string x.

- Reflection: the reflection of a string $x = a_1 a_2 \dots a_h$ is:

  $$x^R = a_h a_{h-1} \dots a_1$$

  The following identities are immediate:

  $$(x^R)^R = x \quad (xy)^R = y^R x^R \quad \varepsilon^R = \varepsilon$$

- Repetition: the m-th power x^m of a string x is the concatenation of x with himself $m-1$ times. The formal definition is the following:

  $$x^m = x^{m-1}x \;\; form \geq 1 \quad x^0 = \varepsilon$$

Repetition and reflection take precedence over concatenation.

Operations are typically defined on a language by extending the string operation to all its phrases:

- Reflection: the reflection L^R of a language L is the finite set of strings that are the reflection of a sentence of L:

  $$L^R = \{x | \exists y \left(y \in L \wedge x = y^R\right)\}$$

- Prefixes: the set of prefixes of a language L is defined as:

  $$Prefixes(L) = \{y | y \neq \varepsilon \wedge \exists x \exists z \left(x \in L \wedge x = yx \wedge z \neq \varepsilon\right)\}$$

  A language is prefix-free if none of the proper prefixes of its sentences is in the language.

- Concatenation: given languages $L^{'}$ and $L^{''}$ we have that concatenation is defined as:

  $$L^{'}L^{''} = \{xy | x \in L^{'} \wedge y \in L^{''}\}$$

- Repetition is redefined as:

  $$L^m = L^{m-1}L \, for \, m \geq 1 \quad L^0 = \{\varepsilon\}$$

  The identity now became:

  $$\varnothing^0 = \{\varepsilon\} \quad L.\varnothing = \varnothing.L = \varnothing \quad L.\{\varepsilon\} = \{\varepsilon\}.L = L$$

1.2 Regular expressions and languages