CSE-309
Theory of Computation

Lecture 01

Introduction to Theory of Computation

Course Overview

Objective : To learn about to how to solve computational problems using necessary theories and models.

Readings : Lecture materials and reference book.

- Marks Distribution(20+30)
    - Tutorials-15
    - Attadance-5
    - Final-30
- Tutorials: There will be 3 tutorials
- Final : Comprehensive closed-book

Reference Books

1. Introduction to the Theory of Computation
   - By Michael Sipser

2. Introduction to Automata Theory, Languages, and Computation
   - By Jeffrey Ullman and John Hopcroft

Introduction to Theory of Computation

In theoretical computer science and mathematics, the **theory of computation** is the branch that deals with how efficiently problems can be solved on a model of computation, using an algorithm.

The field is divided into three major branches:

- automata theory and languages,
- computability theory, and
- computational complexity theory,

The question we will try to answer in this course is:

- "What can be computed?
- What Cannot be computed?
- and where is the line between the two?"

Computational Model

A **Computational Model** is a mathematical object (Defined on paper) that enables us to reason about computation and to study the properties and limitations of computing.

We will deal with Three principal computational models in increasing order of **Computational Power**.

Computational Model

We will deal with three principal models of computations:

1. Finite Automaton (in short FA).
   recognizes Regular Languages .

2. Stack Automaton.
   recognizes Context Free Languages .

3. Turing Machines (in short TM).
   recognizes Computable Languages .

Alphabets and Strings

An alphabet is a set of symbols

Example Alphabet: $\Sigma = \{a, b\}$

A string is a sequence of symbols from the alphabet

Example Strings

a

ab

$abba$

$aaabbbaabab$

$u = ab$

$v = bbbaaa$

$w = abba$

Decimal numbers alphabet $\Sigma = \{0,1,2,\ldots,9\}$

102345 567463386

Binary numbers alphabet $\Sigma = \{0,1\}$

100010001 101101111

Unary numbers alphabet $\Sigma = \{1\}$

Unary number:	1	11	111	1111	11111
Decimal number:	1	2	3	4	5

String Operations

$w = a_1 a_2 \cdots a_n$ $\qquad$ $abba$

$v = b_1 b_2 \cdots b_m$ $\qquad$ $bbbaaa$

Concatenation

$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m$ $\qquad$ $abbabbbaaa$

String Operations

$w = a_1 a_2 \cdots a_n$ $\qquad$ $ababaaabbb$

Reverse

$w^R = a_n \cdots a_2 a_1$ $\qquad$ $bbbaaababa$

String Length

$$w = a_1 a_2 \cdots a_n$$

Length:

$$|w| = n$$

Examples:

$$|abba| = 4$$

$$|aa| = 2$$

$$|a| = 1$$

Length of Concatenation

$$|uv| = |u| + |v|$$

Example:

$$u = aab, \quad |u| = 3$$

$$v = abaab, \quad |v| = 5$$

$$|uv| = |aababaab| = 8$$

$$|uv| = |u| + |v| = 3 + 5 = 8$$

Empty String

A string with no letters is denoted: λ or ε

Observations:

$$|\varepsilon| = 0$$

$$\varepsilon w = w\varepsilon = w$$

$$\varepsilon abba = abba\varepsilon = ab\varepsilon ba = abba$$

Substring

Substring of string:

a subsequence of consecutive characters

String	Substring
$\underline{ab}bab$	ab
$\underline{abba}b$	$abba$
$ab\underline{b}ab$	b
$a\underline{bbab}$	$bbab$

Prefix and Suffix

$abbab$

Prefixes	Suffixes
ε	$abbab$
a	$bbab$
ab	bab
abb	ab
$abba$	b
$abbab$	ε

$$w = \underset{\text{prefix}}{u}\,\underset{\text{suffix}}{v}$$

Another Operation

$$w^n = \underbrace{ww \cdots w}_{n}$$

Example:

$$(abba)^2 = abbaabba$$

Definition:

$$w^0 = \varepsilon$$

$$(abba)^0 = \varepsilon$$

The * Operation

$\Sigma *$: the set of all possible strings from alphabet Σ

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

The + Operation

Σ^{+} : the set of all possible strings from alphabet Σ except ε

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

$$\Sigma^{+} = \Sigma^* - \varepsilon$$

$$\Sigma^{+} = \{a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

Language: a set of strings

String: a sequence of symbols from some alphabet

Example:

Strings: cat, dog, house

Language: {cat, dog, house}

Alphabet: $\Sigma = \{a, b, c, \ldots, z\}$

Languages are used to describe computation problems:

$$PRIMES = \{2,3,5,7,11,13,17,\ldots\}$$

$$EVEN = \{0,2,4,6,\ldots\}$$

Alphabet: $\Sigma = \{0,1,2,\ldots,9\}$

Languages

A language over alphabet Σ is any subset of Σ^*

Examples:

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$$

Language: $\{\varepsilon\}$

Language: $\{a, aa, aab\}$

Language: $\{\varepsilon, abba, baba, aa, ab, aaaaaa\}$

More Language Examples

Alphabet $\Sigma = \{a, b\}$

An infinite language $L = \{a^n b^n : n \geq 0\}$

$$\left.\begin{array}{l} \varepsilon \\ ab \\ aabb \\ aaaaabbbbb \end{array}\right\} \in L$$

$$abb \notin L$$

Prime numbers

Alphabet $\Sigma = \{0,1,2,\boxtimes\ ,9\}$

Language:

$$PRIMES = \{x : x \in \Sigma^{*} \text{ and } x \text{ is prime}\}$$

$$PRIMES = \{2,3,5,7,11,13,17,\boxtimes\ \}$$

Even and odd numbers

Alphabet $\Sigma = \{0,1,2,\boxtimes\ ,9\}$

$EVEN = \{x : x \in \Sigma^{*} \text{ and } x \text{ is even}\}$

$EVEN = \{0,2,4,6,\boxtimes\ \}$

$ODD = \{x : x \in \Sigma^{*} \text{ and } x \text{ is odd}\}$

$ODD = \{1,3,5,7,\boxtimes\ \}$

Unary Addition

Alphabet: $\Sigma = \{1, +, =\}$

Language:

$$ADDITION = \{x + y = z \;:\; x = 1^n, y = 1^m, z = 1^k, \; n + m = k\}$$

$$11 + 111 = 11111 \in ADDITION$$

$$111 + 111 = 111 \notin ADDITION$$

Squares

Alphabet: $\Sigma = \{1, \#\}$

Language:

$$SQUARES = \{x\#y \ : x = 1^n, y = 1^m, m = n^2\}$$

$$11\#1111 \in SQUARES$$

$$111\#1111 \notin SQUARES$$

Note that:

Sets $\emptyset = \{\ \} \neq \{\varepsilon\}$

Set size $|\{\ \}| = |\emptyset| = 0$

Set size $|\{\varepsilon\}| = 1$

String length $|\varepsilon| = 0$

Operations on Languages

The usual set operations

$$\{a, ab, aaaa\} \cup \{bb, ab\} = \{a, ab, bb, aaaa\}$$

$$\{a, ab, aaaa\} \cap \{bb, ab\} = \{ab\}$$

$$\{a, ab, aaaa\} - \{bb, ab\} = \{a, aaaa\}$$

Complement:

$$\overline{L} = \Sigma^* - L$$

$$\overline{\{a, ba\}} = \{\varepsilon, b, aa, ab, bb, aaa, \ldots\}$$

Reverse

Definition: $L^R = \{w^R : w \in L\}$

Examples: $\{ab, aab, baba\}^R = \{ba, baa, abab\}$

$$L = \{a^n b^n : n \geq 0\}$$

$$L^R = \{b^n a^n : n \geq 0\}$$

Concatenation

Definition:

$$L_1L_2 = \{xy : x \in L_1, y \in L_2\}$$

Example:

$$\{a, ab, ba\}\{b, aa\}$$

$$= \{ab, aaa, abb, abaa, bab, baaa\}$$

Another Operation

Definition: $L^n = \underbrace{LL\cdots L}_{n}$

$$\{a,b\}^3 = \{a,b\}\{a,b\}\{a,b\} =$$

$$\{aaa,aab,aba,abb,baa,bab,bba,bbb\}$$

Special case:

$$L^0 = \{\varepsilon\}$$

$$\{a,bba,aaa\}^0 = \{\varepsilon\}$$

Star-Closure (Kleene *)

All strings that can be constructed from L

Definition:

$$L^* = L^0 \cup L^1 \cup L^2 \cdots$$

Example:

$$\{a,bb\}^* = \left\{\begin{array}{l} \lambda, \\ a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{array}\right\}$$

Positive Closure(Kleene +)

Definition: $L^{+} = L^{1} \cup L^{2} \cup \cdots$

Same with L^{*} but without the λ

$$\{a,bb\}^{+} = \left\{ \begin{array}{l} a,bb, \\ aa,abb,bba,bbbb, \\ aaa,aabb,abba,abbbb,\ldots \end{array} \right\}$$