جامعة الشيخ الطوسي

ALSHEIKH ALTOOSI UNIVERSITY

Theory of Computation

النظرية الاحتسابية

المرحلة الثانية

2025-2024

م.م هناء علي

hana.al.shaibani@altoosi.edu.iq

Lecture One
Introduction

Computation: is simply a sequence of steps that performed by computer.

Computation Theory: is the branch that deals with how efficiently problems can be solved on a model of computation, using an algorithm. This field is divided into three major branches:

1- **Automata theory:** Automata Theory deals with definitions and properties of different types of "computation models". Examples of such models are:
- Finite Automata: These are used in text processing, compilers, and hardware design.
- Context-Free Grammars: These are used to define programming languages and in Artificial Intelligence.
- Turing Machines: These form a simple abstract model of a "real" computer, such as your PC at home.

2- **Computability theory:** Computability theory deals primarily with the question of the extent to which a problem is solvable on a computer. In other words, classify problems as being solvable or unsolvable.

3- **Complexity theory:** Complexity theory considers not only whether a problem can be solved at all on a computer, but also how efficiently the problem can be solved. Two major aspects are considered:
- Time complexity: and how many steps does it take to perform a computation.
- Space complexity: how much memory is required to perform that computation.

<u>Some Applications of Computation Theory:</u>

1. Design and Analysis of Algorithms.
2. Computational Complexity.
3. Logic in Computer Science.
4. Compiler.
5. Cryptography.
6. Randomness in Computation.
7. Quantum Computation

Sets

A set is a collection of "objects" called the elements or members of the set.

Common forms of describing sets are:

- List all elements, e.g. {a, b, c, d}.
- Form new sets by combining sets through operators.

Examples in Sets Representation:

- C = { a, b, c, d, e, f }   finite set
- S = { 2, 4, 6, 8, …}   infinite set
- S = { j : j > 0, and j = 2k for k > 0 }
- S = { j : j is nonnegative and even }

Terminology and Notation:

- To indicate that x is a member of set S, we write $x \in S$.
- To denote the empty set (the set with no members) as {} or $\emptyset$.
- If every element of set A is also an element in set B, we say that A is a subset of B, and write $A \subseteq B$ or $B \supseteq A$.
- If A is not a part of B, if at least one of the elements of A does not belong to B then we say that A is not a subset of B, and write $A \nsubseteq B$ or $B \nsupseteq A$.

Basic Operations on Sets:

- **Complement:** $\acute{A}$ or $\bar{A}$ or A^c
  $\bar{A} = \{ x : x \notin A, x \in U\}$
  Contain all elements in universal set which are not in A.
- **Union:** consist of all elements in either A or B
  $A \cup B = \{ x : x \in A \text{ or } x \in B\}$
- **Intersection:** consist of all elements in both A or B $A \cap B = \{ x : x \in A \text{ and } x \in B\}$
- **Difference (/):** consist of all elements in A but not in B $A / B = \{ x : x \in A \text{ but } x \notin B\}$

-

Properties of Sets:

Let A, B, and C be subsets of the universal set U.

- **Distributive properties**

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

- **Idempotent properties**

$A \cap A = A.$

$A \cup A = A.$

- **Double Complement property**

$(A^{\sim})^{\sim} = A.$

- **De Morgan's laws**

$A \cup B)^{\sim} = A^{\sim} \cap B^{\sim}$
$(A \cap B)^{\sim} = A^{\sim} \cup B^{\sim}$

- **Commutative properties**

$A \cap B = B \cap A.$
$A \cup B = B \cup A.$

- **Associative laws**

$A \cap (B \cap C) = (A \cap B) \cap C$
$A \cup (B \cup C) = (A \cup B) \cup C$

- **Identity properties**

$A \cup \emptyset = A$
$A \cap U = A$

- **Complement properties**

$A \cup A^{\sim} = U$
$A \cap A^{\sim} = \emptyset$

Language

Symbols: Symbols are an entity or individual objects, which can be any letter, alphabet, or any picture.

Example:
1, a, b, #

Alphabets: Alphabets are a finite set of symbols. It is denoted by ∑.

Examples:
∑ = {a, b}
∑ = {A, B, C, D}
∑ = {0, 1, 2}
∑ = {#, β, Δ}

String: It is a finite collection of symbols from the alphabet. The string is denoted by w.

Example:
If ∑ = {a, b}, various string that can be generated from ∑ are {ab, aa, aaa, bb, bbb, ba, aba, ...}.

✍ A string with no symbols is known as an ***empty string***. It is represented by epsilon (ϵ) or lambda (λ) or null (Λ).

✍ The number of symbols in a string w is called the ***length of a string***. It is denoted by |w|.

Example: w
= 010
|w| = 3
|00100| = 5
|ab| = 2

| Λ | = 0

Language: A language is a set of strings of terminal symbols derivable from alphabet. A language which is formed over Σ can be Finite or Infinite.

Example:

a) L1 = {Set of string of length 2}
= {aa, bb, ba, bb} **Finite Language**

b) L2 = {Set of all strings starts with 'a'}
= {a, aa, aaa, abb, abbb, ababb, …} **Infinite Language**

Types of Languages:

1- ***Natural Languages:*** They are languages that spoken by humans e.g.: English, Arabic and France. It has alphabet: ∑={a, b, c, …. z}. from these alphabetic we make sentences that belong to the language.

2- ***Programming Language:*** (e.g.: c++, Pascal) it has alphabetic: ∑={a, b, c, z, A, B, C, .. , Z , ?, /, -, \}. From these alphabetic we make sentences that belong to programming language.

<u>Example:</u>

Alphabetic: ∑= {0, 1}.
Sentences: 0000001, 1010101

<u>Example:</u>

Alphabetic: ∑= {a, b}.
Sentences: ababaabb, bababbabb

<u>Example:</u>

Let ∑ = {x} be set of alphabet of one letter x. we can write this in form:

L_1 = {x, xx, xxx, …} or write
this in an alternate form: L_1 =
{x^n for n = 1, 2, 3, ...}

Let a = xxx and b = xxxxx
Then ab = xxxxxxxx = x^8
ba = xxxxxxxx = x^8

<u>Example:</u>

L_2 = { x, xxx, xxxxx, ... }
= { x^{odd}}
= { x^{2n+1} for n = 0, 1, 2, 3, ... }

PALINDROME

Let us define a new language called **PALINDROME** over the alphabet
$\sum$= {a, b}
PALINDROME = { Λ, and all strings x such that reverse(x) = x } If we begin listing the elements in PALINDROME we find:
PALINDROME = { Λ, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba, ... }

Kleene Closuer

They are two repetition marks, also called Closuer or Kleene Star.
* : Repeat (0 – n) times.
\+ : Repeat (1 – n) times.

Example:

If $\sum$ = {x}, then

$\sum^*$ = L_3 = { Λ, x, xx, xxx, …}
$\sum^+$ = L_3 = { x, xx, xxx, …}

Example:

If $\sum$ = {0, 1}, then

$\sum^*$ = L_4 = { Λ, 0, 11, 001, 11010, …}
$\sum^+$ = L_4 = { 0, 01, 110, 101, …}

Example:

If $\sum$ = {aa, b}, then

$\sum^*$ = L_5 = { Λ, aab, baa, baab, aabb, …}
$\sum^+$ = L_5 = { aaaa, b, baaaa, bb, …}

✍ في هذه اللغة الكلمة (ab) غير مقبولة لأن (aa) هو حرف واحد ولايجوز تجزئته.

Example:

If $\sum = \{\ \}$, then

$\sum^* = L_4 = \{\Lambda\}$

$\sum^+ = L_4 = \emptyset$ or $\{\ \}$

Lecture Two

Regular Expression (RE)

Regular languages are formal languages that can be expressed using regular expressions.

Regular languages can be generated from one-element languages by applying certain standard operations a finite number of times. These simple operations include (concatenation, union, and Kleen closure).

Regular expressions can be thought of as the algebraic description of a regular language. Regular expression can be defined by the following rules:

1. Every letter of the alphabet ∑ is a regular expression.
2. Null string ʌ and empty set ∅ are regular expressions.
3. If r1 and r2 are regular expressions, then
   (i) r1, r2
   (ii) r1r2 (concatenation of r1r2)
   (iii) r1 + r2 (union of r1 and r2)
   (iv) r1*, r2* (kleen closure of r1 and r2) are also regular expressions
4. If a string can be derived from the rules 1, 2 and 3 then it is also a regular expression.

Note that a* means zero or more occurrence of a in the string while a^+ means that one or more occurrence of a in the string. That means a* denotes language L = {ʌ , a, aa, aaa, ….} and a^+ represents language L = {a, aa, aaa, ….}. And also note that there can be more than one regular expression for a given set of strings.

<u>Example:</u> Write the language for each of the following regular expressions, ∑ = {a,b}.

1- (ab)* = {ʌ, ab, abab, ababab, … }

2- ab*a = {aa, aba, abba, abbba, abbbba, …}

3- a*b* = {ʌ, a, b, aa, ab, bb, aaa, aab, abb, bbb, aaaa, …}

✍Notice that **ba** and **aba** are <u>not</u> in this language. Also we should be very careful to observe that **a*b* ≠ (ab)***

Example: Write a regular expression for the language containing odd number of 1s, $\sum = \{0,1\}$.

The language will contain at least one 1. It may contain any number of 0s anywhere in the string. So the language we have to write a regular expression for is 1, 01, 01101, 0111, 111, … This language can be represented by the following regular expression:

$$0^*(10^*10^*)^*10^*$$

Example: Write the language for each of the following regular expressions, $\sum = \{x\}$.

1- $L_1 = \{x^{odd}\} = x(xx)^*$ **or** $(xx)^*x = \{x, xxx, xxxxx, \ldots\}$ 2- $L_2 =$

$\{x^{even}\} = (xx)^*$ **or** $(xx)^*xx$ **or** $xx(xx)^* = \{\wedge, xx, xxxx, \ldots\}$

Examples:

1- Consider the language L_3 defined over the alphabet $\sum = \{a, b, c\}$, All the words in L_3 begin with an **a** or **c** and then are followed by some number of **b's**. We may write this as:

$$(a + c)b^*$$

2- Consider a finite language L_4 that contains all the strings of **a's** and **b's** of length exactly three.

$$L_4 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

So we may write:

$$(a + b)(a + b)(a + b) \quad \textbf{or} \quad (a + b)^3$$

In general, if we want to refer to the set of all possible strings of **a's** and **b's** of any length, we could write:

$$(a + b)^*$$

3- Construct RE for all words that begin with the letter **a**:

$$a(a + b)^*$$

4- All words that begin with an **a** and end with **b** can be defined by the expression: $a(a + b)^*b$

5- The language of all words that have at least two **a's** can be described by the expression:

$$(a + b)^*a(a + b)^*a(a + b)^*$$

6- The language of all words that have at least one **a** and at least one **b**:

$$(a + b)^*a(a + b)^* b(a + b)^* \quad \underline{\textbf{or}} \quad bb^*aa^*$$

7- The words of the form some **b's** followed by some **a's**. These exceptions are all defined by the regular expression: $bb^*aa^* \equiv b^+a^+$

Example: Write a regular expression for the language

$L = \{ab^n w: n >= 3, w \in (a + b)^+\}$

The strings in the language begins with a followed by three bs and followed by string w. w will contain at least one a or b. The strings are like abbba, abbbb, abbbbababab, abbbaaaa, . . . This language can be represented by the following regular expression

$$\mathbf{ab^3(a + b)^+}$$

Homework:

1- Find a regular expression over the alphabet {a, b}:

a. L_1 = {All strings that contain exactly three a's}
b. L_2 = {All strings that end with ab}
c. L_3 = {All strings in which letter a is even number}

2- Find the output (words) for the following regular expressions:

a. aa^*b
b. $(a + b)^*ba$
c. $(11 + 0)^*(0+11)^*$