Introduction to Computer Theory (Lec1)

what are computation theory?

It is the theory of computer or what is the computer dose, it necessary in the teaching of courses on computer design, Artificial Intelligence, the analysis of algorithms, and so. Of all the programming skills undergraduate students learn, two of the most important are the abilities to recognize and manipulate context-free grammars and to understand the power of the recursive interaction of parts of a procedure. Very little can be accomplished if each advanced course has to begin at the level of defining rules of production and derivations.

In computation theory we analyses every procedures in computer and study the algorithm.

Algorithm: is a collection of rules to solve some problems.

Some Basic Mathematics

1- "Symbol": is an abstract entity such as point and line in geometry.
   Ex: letters a...z
   digits 0..9

2- "String" Or "Word": is define as sequence of symbols.
   Ex: if symbols (a,b,c) then (abc) is a string.

3- "Empty String": is a string consisting of zero symbols and denoted by (λ)

4- "prefix": is a string of any numbers that leading symbols of the string.
   Ex: a b c , has prefix (a): λ, a, ab, abc,.....

5- "Suffix": is any number of trailing symbols.
   Ex: a b c , has suffix (c): λ, c, bc, abc,.....

6- "Concatenation" of two strings is the string formed by northing the first followed by the second with no intervenes space.
   Ex: if we have the two strings (day) and (house) the concatenation of them is (dayhouse).

7- "Alphabet": is a finite set of symbols and denoted by Σ
   Ex: Σ = {a, b} Σ = {a, b, c, d}
   Σ = {0,1} Σ = {1,2,3,...,7}

8- "Formal Language": is a set of strings of symbols from some Alphabet
Definitions

1- we define the "Length of string" to be the number of letters in a string
   we write this function using \(||

Ex: if \(s = \text{abc}\) Length of \(s\) \(|s| = 3\)
   If \(s = 1014\) \(|s| = 4\)
   \(|\lambda| = 0\)

2- let us introduce two function "reverse" if \((a)\) is a word in some
   language \((L1)\) then \(\text{reverse}(a)\) is the same string of letters spelled back
   word called \(\text{reverse} \) of \((a)\), even if this back word string is not a word
   in \((L1)\)

   ex: \(\text{reverse}(abc) = cba\)
   \(\text{reverse}(101) = 101\)

3- let us define the new language called "Palindrome" over the alphabet
   \(\Sigma = \{a, b\}\)

   Palindrome = \(\{\lambda, a, b, aa, bb, aab, bab, \ldots\}\)
   Given in alphabet \(\Sigma\) we wish to define along in which any string
   element from \(\Sigma\) is a word even the \((\text{un string})\) (not string) this
   language is called (closure) of the language, it is denoted by writing a
   star after the name of the alphabet as a subscript \(\Sigma^*\)

   \(\Sigma = \{x\}\)
   \(\Sigma^* = \{\lambda, x, xx, xxx, xxxx, \ldots\}\)
   \(\Sigma = \{0, 1\}\)
   \(\Sigma^* = \{\lambda, 0, 1, 00, 01, 11, 000, 001, 010, 011, 111\}\)
   \(\Sigma = \{a, b\}\)
   \(\Sigma^* = \{\lambda, a, b, ab, ba, aa, bb, aab, bbb, \ldots\}\)

   The notation is some time know as the kleen star \(\Sigma^+\) (kleen star is \(\Sigma^*\)
   but without \(\lambda\))

Theorm1: for any set \(S\) of strings we have \(S^* = S^{**}\)
H.W : prove that \( L^* = L^{**} \)

Note\ K.W is the words know by the computer and used for fixed procedures
words= \{ var, begin ,......\}
Puctions \{ , , " ", : ,| ,\ , .... \}
Arithmetic operation\{ / , * , - , + \}
Syntax \{ if , then , else \}
Set and Relations of the Set (LEC 2)

Set: is the unordered collection of objects (member of set) without repeat.

Relation of the set

1- listed method

D={sun, mon, tu, wed, thu, fri, sat}
A={11,12,22}
E={2,4,6,8,10,...} {infinite set}
C={3, 5,7,......}

2- formal method

D={x: x € day of week}
A={x: x is a two digit integer of whose digit is 1 or 2}
E={x: x even no.}
C={x: x is odd no. greater than 1}

- Empty set: the set contained no element {} or \( \emptyset \)

- Sub set: sub set of set we say a set of A is sub set of B if written \( A \subseteq B \) if written every element of A is an element of the set B

\{1,2,4\} \subseteq \{1,2,3,4\}
\{2,4,6\} \subseteq \{1,2,3,4\}

- Tow set are equal if they have exactly the same element

\( A = B \) if \( A \subseteq B \) and \( B \subseteq A \)
A={1,2,3,4}
B={2,4,3,1}

Empty set: is sub set of every set and subset of itself
Basic operation on set

1- **Complement** (negative)
   \[ A^\sim = \{ x : x \not\in A \} \] consists of all elements in the universe which are not in A

2- **Union** (\( \cup \))
   \[ A \cup B = \{ x : x \in A \text{ or } x \in B \} \]
   Consist all elements in either A or B

3- **Intersection** (\( \cap \))
   \[ A \cap B = \{ x : x \in A \text{ and } x \in B \} \]
   Consist all elements in both A and B

4- **Difference** (\( \setminus \))
   \[ A \setminus B = \{ x : x \in A \text{ and } x \not\in B \} \]
   Consist all elements in A and not in B
- Cartesian product
A_1 \times A_2 = \{ (a_1, a_2) : (a_1 \in A_1, a_2 \in A_2) \}
Ex: A_1 = \{1, 0\} \quad A_2 = \{x, y, z\}
A_1 \times A_2 = \{ (0,x) , (0,y) , (0,z) , (1,x) , (1,y) , (1,z) \}

- Power of the set
Power of the set \(2^A\) set of all sub set of A
A = \{1,2\}
\(2^A = \{ O , [1], [2], [1,2] \}\)

- Relations: is any sub set of A_1 \times A_2
where A_1 is called the domain of R and A_2 is called rang of R

Ex: A = \{0, 1, 2, 3\}
L = \{(0,1), (0,2), (0,3), (1,2), (1,3), (2,3)\}
E = \{(0,0), (1,1), (2,2), (3,3)\}
Properties of the relation

1- Reflexive: if

2- Symmetric: if

3- Transitive: if

4- Equivalent: if

- Closure of relation *

If R an arbitrary relation on A then the reflexive, symmetric, and transitive cluster of R is the smallest reflexive, symmetric, and transitive relation on A with R as subset.

Let R={ (0,1), (1,1), (1,2) }
Is a relation on A={0, 1, 2}
Reclusive – closure (R) = {(0,1), (1,1), (1,2), (0,0), (2,2)}
Symmetric - closure (R) = {(0,1), (1,1), (1,2), (1,0), (2,1)}
Transitive - closure (R) = {(0,1), (1,1), (1,2), (0,2)}

H.W: let R={ (1,2), (2,2), (2,3) } be a relationship on the set {1, 2, 3} find the reflexive, symmetric, and transitive closure of R.
Graph Representation (LEC 3)

\((G)\): is denoted by \( G = (V, E) \) consists of a finite set of vertices (Nodes) \( V \) and a set of pairs of vertices \( E \) called edges

Ex: \( V = \{1, 2, 3, 4, 5\} \)
\( E = \{(n, m): n+m=4 \text{ or } n+m=7\} \)
\( E = \{(1,3), (2,2), (3,4), (2,5)\} \)

Direct Graph (digraph): is denoted by \( G = (V, E) \) consists of a finite set of vertices (Nodes) \( V \) and a set of ordered pairs \( E \) called arc
\( G_1 = (V_1, E_1) \)
Ex:

Ex: \( G_2 = (V_2, E_2) \) where \( V_2 = \{v, w, x\} \) and \( E_2 = \{(v,w), (v,x), (w,x), (x,v), (x,x)\} \)
Path: is a sequence of vertices v1,v2,…vk
Where k >= 1 such that vi ---→ vi+1 is an arc
( 1 <= I <= k )

Length of Path: the number of sequence of direct path from V to W of N node
= n-1 or k

Cyclic Path : if V1= Vk

Directed sub graph : the direct sub graph of G is any digraph ( V, E )
where V ( V and E ( E
For the above example ex2: G2(V2,E2)

[v,x]  ( [v,w,x]= V2
[x,v]  ( v2   [w] ( V2
{(v,x) , (x,v) } ( E2 , {(x,v) } ( E2
Disjoint sub Graph:

Tow sub Graph are said to be disjoined if they have no nodes in common in the example a and c are disjoint node
b and c are disjoint node
a and d are joint node

Tree:

Is a digraph with following properties

1- There is one vertex called "Root" with no a predecessor and from which there is a path for every vertex.
2- Each vertex (other than the root) has exactly one predecessor.
3- The successor of each vertex are ordered from left to right.
4- A vertex with no child is called "leaf" the other vertices are called "interiors"
Regular Language and Regular Expression

(RL and RE)(lec 3)

Finite Automata Machine (FA): is a Regular Language that the machine
Let $\Sigma$ be on alphabet $= \text{set of symbols}$
$L_1, L_2, L_3 \text{ be languages from } \Sigma^* = \text{set of string}$
Union $(L_1, L_2) = L_1 + L_2 = \{ x: x \in L_1 \text{ or } x \in L_2 \}$
Concatenation $(L_1, L_2) = L_1 \cdot L_2 = \{ x \cdot y: x \in L_1 \text{ and } y \in L_2 \}$
Kleen- closure $(L) = L^* = \bigcup L_i$
Positive – closure $(L) = L^+ = \bigcup L_i$

Ex:

Three basic operations for constructing new language from existing one

(Union, Concatenation and closure operation) when we start with simples possible language those containing a single string that is either the null string or of length $= 1$, the language that we obtained by using a combination of those operations are that Regular Language.
Such language can therefore be described by an explicit formula involving (Union, Concatenation and closure operation). its common to simplify and replacing the union $(\bigcup)$ by $+$
The result is called the Regular Experian
### Table: Language vs. RE

<table>
<thead>
<tr>
<th>Language</th>
<th>RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ λ }</td>
<td>λ</td>
</tr>
<tr>
<td>{ 0 }</td>
<td>0</td>
</tr>
<tr>
<td>{ 001 }</td>
<td>001</td>
</tr>
<tr>
<td>{ 0,1 }</td>
<td>(0 + 1)</td>
</tr>
<tr>
<td>{ 0,10 }</td>
<td>(0 + 10)</td>
</tr>
<tr>
<td>{ 1, λ } {001}</td>
<td>(1 , λ) 001</td>
</tr>
<tr>
<td>{ 110 }</td>
<td>(110) (0 + 1)</td>
</tr>
<tr>
<td>(0,10)* ({11}* U {001, λ})*</td>
<td>(0+10)* ((11)* +001 + λ)*</td>
</tr>
<tr>
<td>{1}* {10}</td>
<td>1* .10</td>
</tr>
<tr>
<td>{10,11,1101,01}*</td>
<td>(10+11+1101+01)*</td>
</tr>
</tbody>
</table>

### Detention

**Regular Expiration:** A RE over the alphabet, and the corresponding language are defined as follows:

1. \( \emptyset \) is ER, corresponding to empty Language
2. \( \lambda \) is ER, corresponding to empty Language
3. For each symbol \( a \), \( a \) is a RE corresponding to the language \( \{a\} \)
4. For any RE of \( x \), \( a \) is a RE corresponding to the language \( L_r \& L_s \), respectively each of the following is a RE over \( \Sigma \) to the language indicate.
   - (rs) corresponding to \( L_r L_s \)
   - (r+s) corresponding to \( L_r U L_s \)
   - (r)* corresponding to \( L_r^* \)
5. Only those things that can be produced using parts 1-4
6. Along over \( \Sigma \) is RE if there is some RE over \( \Sigma \) corresponding to all
Examples: consider the following REs defined in $\Sigma = \{0,1,2\}$

1- 111 is RE corresponding $L=\{111\}$
2- 0 +1 is RE corresponding $L=\{0,1\}$
3- (0+1) $L=$all string of 0s and 1s including ( $\lambda$, 0,1)
4- (0+1) * 00(0+1)* all string of 0s and 1s and 2 consecutive zero and (0 ,1) at least one
5- (1,10)* all string of 0s beginning with 1 and not has two consecutive 0s
6- (0+1) all string of 0s ending with 1s
7- 0*1*2* any number of 0s follow by any number of 1s follow by any number of 2s
8- 00*11*22* denoted these things in 0*1*2* with at least one of each symbol
9- 00* = 0+ 11* =1+ 22*=2+
Finite Automata machine (FAM) (LEC 4)

Init Automata machine: it is a mathematic model of system with discrete input and output. The system as a whole can in any number of intermediate configuration or states.

DFA (Deterministic finite Automata): Its denoted by 5 order tolls

\[ M = \langle Q, \Sigma, \delta, q_0, F \rangle \]

- \( Q \): is a finite set of internal states
- \( \Sigma \): is a finite set of input alphabet
- \( \delta \): is transition function \( Q(x) \)
- \( q_0 \): \( q_0 \in Q \) is the little status
- \( F \): \( F \subseteq Q \) is an one empty set of finial state

Ex

Accepted String:
A string is said to be accepted by FA if (for \( \delta(q_0, w) \) there exist some \( p \in F \))

\[ \delta(q_0, w) = p \rightarrow p \in F \text{ (w) accepted } \]
\[ p \not\in F \text{ (w) not accepted } \]
Non Deterministic Finite Automata (NFA) (LEC 5)

It is the Finite Automata that allows zero, one or more transition from a state on the same input symbol.

NFA denoted by 5 ordered

\[ <Q, \Sigma, \mathcal{E}, q_0, F> \]

\[ \mathcal{E} = Q^* \Sigma = 2^Q \] any sub set \( <2^Q\)

Ex:

**Accepted string:**

Accepted string = \( \sum W \subseteq \mathcal{E}(q_0, w) F = 0 \)

The Equivalence of DFA and NFA

T.T : Transition Table

T.D. : Transition Diagram
Finite Automata with output (LEC 6)

1- Moore Machine

This machine represent by 6 ordered

\(< \ Q \ , \ \Sigma \ , \ \mathcal{E} \ , \ \lambda \ , q_0 >\)

Where \( Q \ , \ \Sigma \ , \ \mathcal{E} \ , q_0 \) as in DFA

\( \Lambda: Q \) giving the o/p associated with each state.

Ex:

2- Mealy Machine

This machine represent by 6 ordered

\(< \ Q \ , \ \Sigma \ , \ \mathcal{E} \ , \ \lambda \ , q_0 >\)

Where \( Q \ , \ \Sigma \ , \ \mathcal{E} \ , q_0 \) are same in the Moore machine, but the output function \( \lambda \) is denoted by

\( \Lambda: Q \times \Sigma \)

Ex:

Note: the final state in Moor and Mealy Machine, the process is terminated when the last letter is read and the output character is printed.
Context Free Grammar (CFG)  (LEC 7)

Definition: the language generated by CFG is the set of all strings of terminal that can be produced from start symbol using the production as sub situation
The Language generated by CFG is called Contacts Free Language CFL

CFG: consist of the following:
< N , T, P , S >   where
N: Non terminal , one of which  is the symbol S standing for start state
T: Terminal, which are on alphabet of letters that going to make strings that will be the word of along.
P: set of protection of the form
A
Where A   N ,     (N+T)*
S: start state , S  N