COMPUTER ENGINEERING DEPARTMENT

ASSIGNMENT NO-05

Sub: Theory of Computer Science

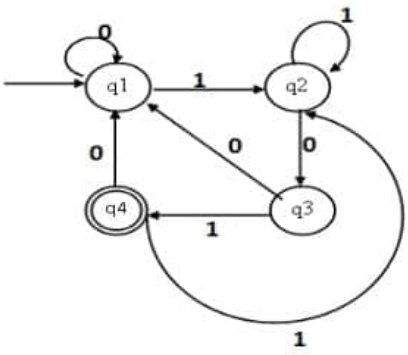
COURSE: T.E. Year: 2020-2021 Semester: V
DEPT: Computer Engineering
SUBJECT CODE: CSC504 DUE DATE: 22/11/2020
==========

Roll No. 50 Name: Amey Thakur

Class: TE-Comps B Date of Submission: 20/11/2020

Tutorial 5

1. Explain the Closure Properties for Regular Languages
2. Find R.E. equivalent to following FA.



3. Write Short note on variants of TM
4. Write Short note on Universal Turing Machine
5. What is TM? Give the power of TM over FSM. Explain the undecidability and incompleteness of TM.
6. State and prove Rice's theorem

Q.1 Explain the closure properties for Regular Languages

Ans:

① Closure under Union

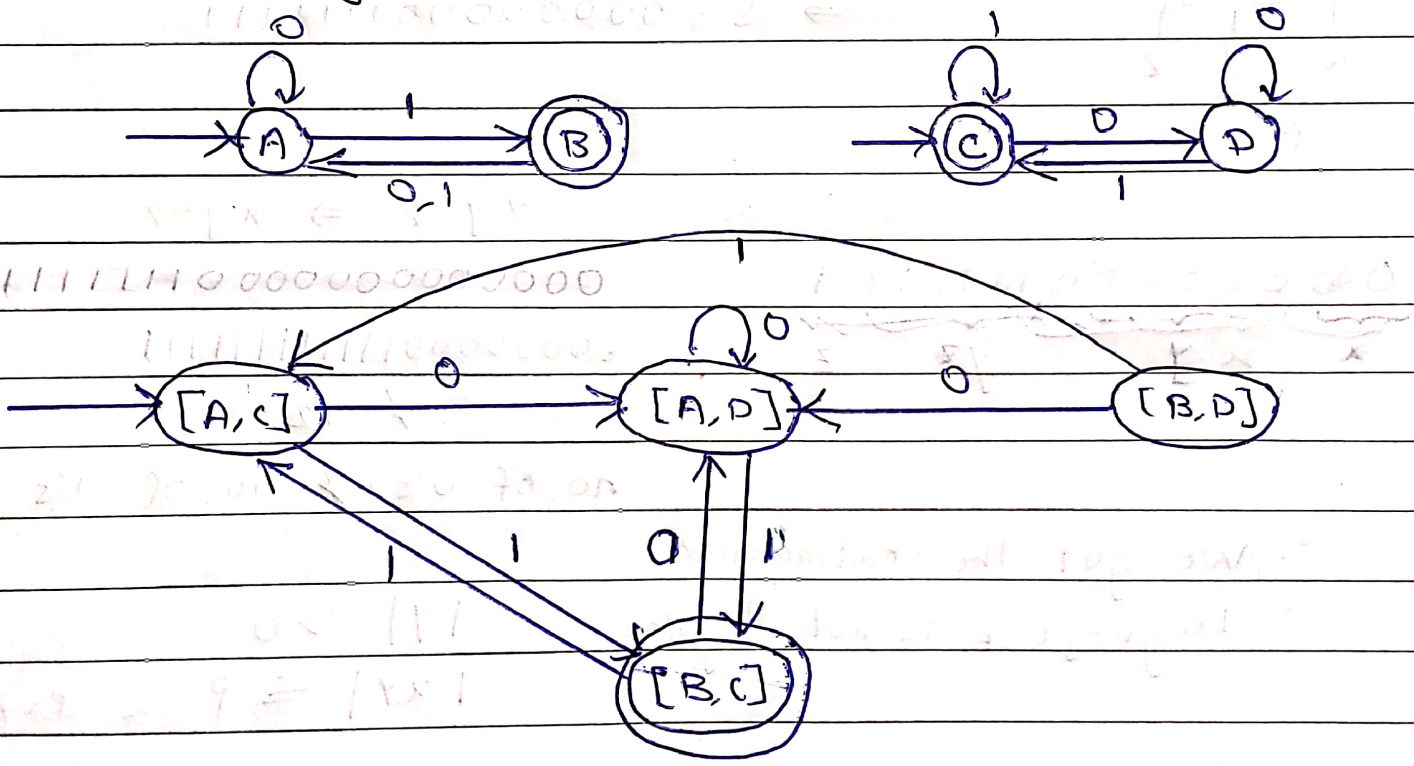
- If L and M are regular languages, so is L ∪ M.
- Proof: Let L and M be the languages of regular expressions R and S respectively.
- Then R + S is regular expression whose language is L ∪ M.

② Closure under Concatenation and Kleene Closure.

- The same idea can be applied using Kleene closure:
- RS is a regular expression whose language is LM.
- R^* is a regular expression whose language is L^*.

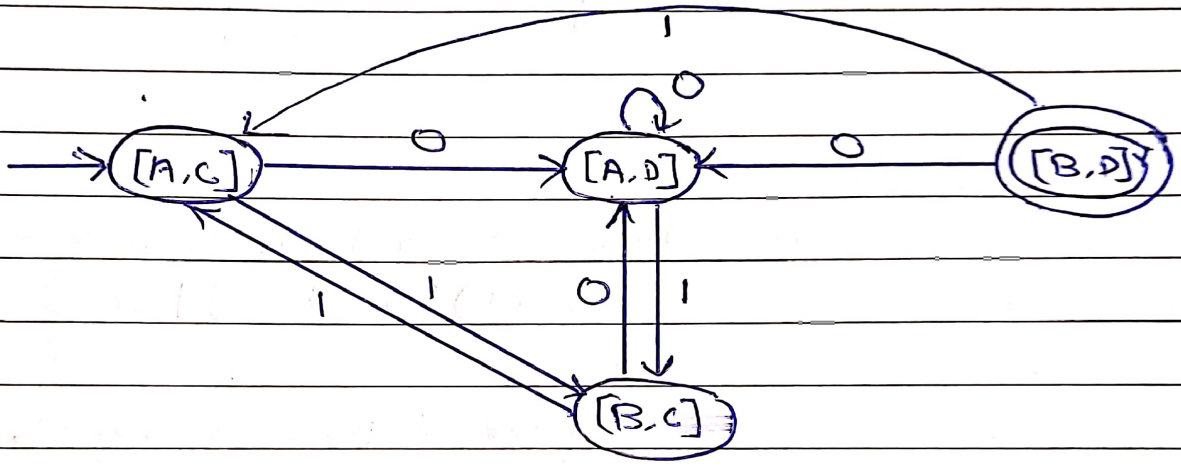
③ Closure under intersection.

- If L and M are regular languages, so is L ∩ M.
- Proof: Let A and B be two DFA's whose regular languages are L and M respectively.
- Now, Construct C, the product automation of A and B. Make the final states of C be the pairs consisting of final states of both A and B.



④ Closure under Difference

- If L and M are regular languages, so is L−M, which means all the strings that are in L, but not in M respectively. Now, Construct C, the product automation of A and B. Make the final states of C be the pairs consisting of final states of A but not B. The DFA's A-B and C-D remain unchanged, but the final DFA varies as follows



⑤ Closure under concatenation

- The complement of language L (with respect to an alphabet Σ such that Σ* contains L) is Σ*− L Since Σ* is surely regular, the complement of a regular language is always regular.

⑥ Closure under Reversal

- Given language L, LR is the set of strings whose reversal is in L
- L = {0, 01, 100}, LR = {0, 10, 001}
- Basis : If E is a symbol a, ε or ϕ then ER = E.
- Induction: If E is
  - F + G, then ER = FR + GR
  - FG, then ER = GRFR
  - F, then ER = (FR)

Let $E = 01^* + 10^*$

$E^R = (01^* + 10^*)^R = (01^*)^R + (10^*)^R$

$= (1^*)^R 0^R + (0^*)^R 1^R$

$= (1^R)^* 0 + (0^R)^* 1$

$= 1^* 0 + 0^* 1$

⑦ Closure under homomorphism.

- Define: A homomorphism on an alphabet is a function that gives string for each symbol in that alphabet.

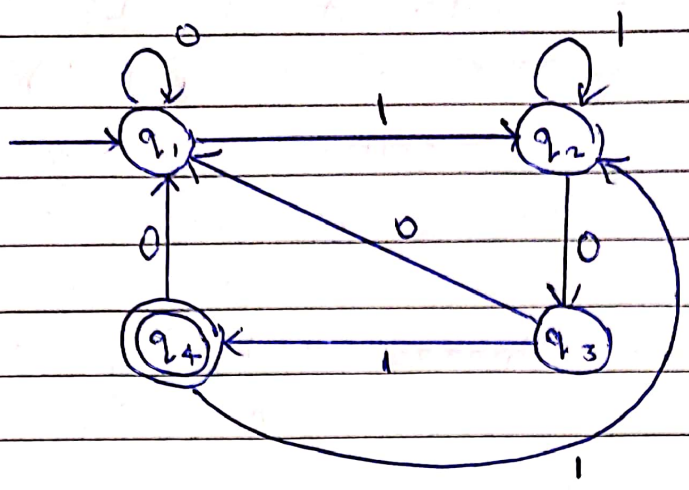
Closure Property

- If L is a regular language and h is homomorphism on its alphabet, then $h(L) = \{h(w) \mid w \text{ is in } L\}$ is also a regular language.
- Proof: Let E be a regular expression for L.
  Apply h to each symbol in E.
  Language of resulting RE is $h(L)$
- Example:
  Let $h(0) = ab$; $h(1) = \varepsilon$
  Let L be the language of regular expression $01^* + 10^*$
  Then $h(L)$ is the language of regular expression $ab\varepsilon^* + \varepsilon(ab)^*$
  $ab\varepsilon^* + \varepsilon(ab)^*$ can be simplified
  $\varepsilon^* = \varepsilon$, so $ab\varepsilon^* = ab\varepsilon$
  ε is the identity under concatenation
  That is, $\varepsilon E = E\varepsilon$ for any RE E.
  Thus, $ab\varepsilon^* + \varepsilon(ab)^* = ab\varepsilon + \varepsilon(ab)^*$
  $= ab + (ab)^*$
  Finally, $L(ab)$ is contained in $L((ab))$,
  so a RE for $h(L)$ is (ab)

⑧ Closure under inverse homomorphism

- A. Start with a DFA A for L.
- B. Construct a DFA B for $h^{-1}(L)$ with :-
  - The same set of states.
  - → The same start state
  - → The same final state
  - → Input alphabet = the symbol to which homomorphism h applies.

Q2. Find R.E. equivalent to following F.A.



Ans:

(A) Step 1:

Expressing the automata as a set of equation.

$q_1 = q_1 0 + q_3 0 + q_4 0 + \epsilon$ — (1)

$q_2 = q_1 1 + q_0 1 + q_4 1$ — (2)

$q_3 = q_2 0$ — (3)

$q_4 = q_3 1$ — (4)

(B) Step 2:

Solving the set of equations.

Putting the values of q_3 and q_4 (From (3) & (4)) in equation (1) & (2)

$q_1 = q_1 0 + (q_2 0) 0 + (q_3 1) 0 + \epsilon$

$= q_1 0 + q_2 00 + q_2 010 + \epsilon$

$= q_1 0 + q_2 (00 + 010) + \epsilon$ — (5)

$q_2 = q_1 1 + q_2 1 + (q_3 1) 1$

$= q_1 1 + q_2 1 + q_3 11$

$= q_1 1 + q_2 1 + q_2 011$

$= q_1 1 + q_2 (1 + 011)$ — (6)

From equation (6)

$q_2 = q_1 1 + (1 + 011)^*$ — (7)

From equation (5) and (7)

$q_1 = q_1 0 + q_1 1 (1 + 011)^* (00 + 010) + \epsilon$

$q_1 = q_1 [0 + 1 (1 + 011)^* (00 + 010)] + \epsilon$

$\therefore q_1 = [0 + 1 (1 + 011)^* (00 + 010)]^* \quad — (8)$

From equation (7) and (8)

$q_2 = [0 + 1 (1 + 011)^* (00 + 010)]^* 1 (1 + 011)^* \quad — (9)$

Now,

$q_4 = q_3 1$

$q_4 = q_2 01$

$q_4 = [0 + 1 (1 + 011)^* (00 + 010)]^* 1 (1 + 011)^* 01$

Which is the required R.E.

Q.3. Write short note on variants of TM.

Ans:

① Multiple track Turing Machine.

- A k- track turing machine (for some $k > 0$) has k-tracks and one R/W head that reads and writes all of them one by one.
- A k- track turing machine can be simulated by a single track turing machine.

② Two-way infinite Tape Turing Machine.

- Infinite tape of two way infinite tape turing machine is unbounded in both directions left and right.
- Two way infinite tape turing machine can be simulated by one way infinite turing machine (standard turing machine)

③ Multi-tape turing machine.

- It has multiple tapes and controlled by a single head.
- The multitape turing machine is different from K-track turing machine but expressive power is same
- Multitape Turing Machine can be simulated by Single Tape Turing Machine.

④ Multi-Tape multi-head Turing Machine:

- The Multi-tape Turing Machine has multiple heads and multiple tapes.
- Each tape controlled by separate head.
- Multi-tape Multi-head Turing machine can be simulated by standard Turing machine.

⑤ Multi - dimensional Tape Turing Machine

- It has multi-dimensional tape where head can move any direction that is left, right, up or down.
- Multi-dimensional tape Turing machine can be simulated by one-dimensional Turing Machine.

⑥ Multi-head Turing Machine

- A multi-head turing machine contain two or more heads to read the symbols on the same tape.
- In one step all the heads sense the scanned symbols and move or write independently.
- Multi-head turing machine can be simulated by single head Turing machine.

⑦ Non - deterministic Turing Machine

- A non-deterministic Turing machine has a single one way infinite tape
- For a given state and input symbol has at least one choice to move (infinite numbes of choices for the next move), each choice
- several choices of path that it might follow for a given input string.
- A non-deterministic Turing machine is equivalent to deterministic Turing Machine

Q.4 Write Short Note on Universal Turing Machine

Ans:

- A Turing Machine is said to be universal turing machine if it can accept:
  - → The input data
  - → An algorithm (description) for computing
- This is precisely what a general purpose digital computer does. A digital computer accepts a program written in High Level Language. Thus, a general purpose Turing Machine will be called a Universal Turing Machine if it is powerful enough to simulate the behaviour of any digital computer, including any Turing machine itself.
- More precisely, a universal turing machine can simulate the behaviour of an arbitrary turing machine over any set of input symbols. Thus it is possible to create a single machine that can be used to compute any computable sequence.

If this machine is supposed to be supplied with the tape of the beginning of which is written in the input string of quintuple separated with some special symbol of some computing machine M, then the universal turing machine U will compute the same strings as those by M.

- The model of a universal turing machine is considered to be a theoretical breakthrough that led to the concept of stored programmer computing device.

- Designing a general purpose turing machine is a more complex task. Once the transmission of turing machine is defined, the machine is restricted to carrying out one particular type of computation.
- Digital computers, on the other hands, are general purpose machines that cannot be considered equivalent to general purpose digital computers until they are designed to be reprogrammed.
- By modifying our basic model of a Turing machine we can design a universal turing machine. The modified turing machine must have a large number of states for stimulating even a simple behaviour. We modify our basic model by:
  - → Increase the number of read/write heads
  - → Increase the number of dimensions of input tape.
  - → Adding a special purpose memory.
- All the above modifications in the basic model of a turing machine will almost speed up the operations of the machine can do.
- A number of ways can be used to explain to show that turing machines are useful model of real computers. Anything that can be computed by a real computer can also be computed by a turing machine. A turing machine, for example can simulate any type of functions used in programming language. Recursion and parameter passing are some typical examples. A turing machine can also be used to simplify the statements of an algorithm.
- A turing machine is not very capable of handling it in a given finite amount of time. Also, turing machines are not designed to receive unbounded input as many real programmers like word processor, operating system and other system softwares.

Q.5 What is TM & Give the power of TM over FSM Explain the undecidability and incompleteness of TM.

Ans:

Turing Machine

A Turing Machine is an accepting device which accepts the languages (recursively enumerable set) generated by type 0 grammars. It was invented in 1936 by Alan Turing.

- A Turing Machine (TM) is a mathematical model which consists of an infinite length tape divided into cells on which input is given. It consists of a head which reads the input tape. A state register stores the state of a Turing Machine. After reading an input symbol, it is replaced with another symbol, its internal state is changed and it moves from one cell to the right or left. If the Turing Machine reaches the final state, the input string is accepted, otherwise rejected.

- A Turing machine can be formally described as a 7 - tuple $(Q, X, \Sigma, \delta, q_0, B, F)$ where,

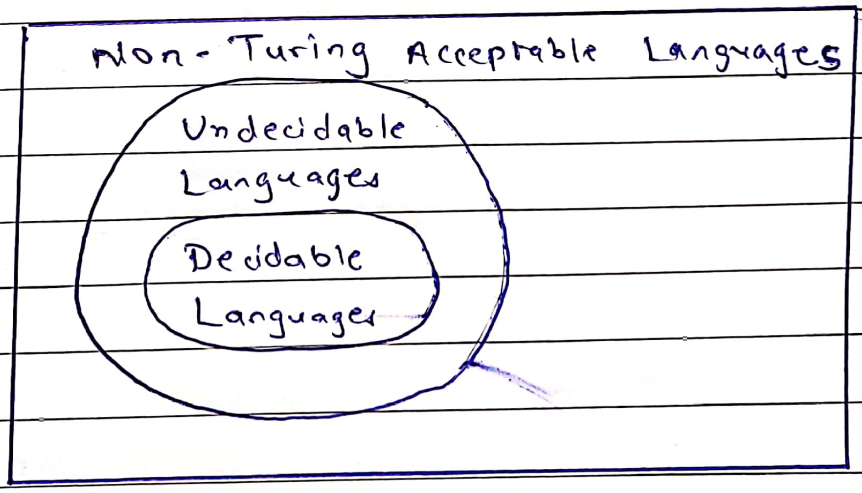
1. Q is a finite set of states
2. X is the tape alphabet.
3. Σ is the input alphabet
4. δ is transition function,
   $\delta : Q \times X \longrightarrow Q \times X \times \{Left\_shift, Right\_shift\}$.
5. q_0 is the initial state
6. B is the blank symbol
7. F is the set of final states.

Power of Turing Machine over FSM.

- TM is more powerful than FSM.
- FSM has no memory; TM has additional memory in the form of a tape.
- FSM cannot modify its input. A TM can modify its own input
- FSM cannot be used for arithmetic operations. A TM can perform arithmetic operations
- A language accepted by FSM is RL. It cannot handle CFL, Recursive languages. These languages can be handled by TM.

Undecidability and incompleteness of TM.

- For an undecidable language, there is no Turing Machine which accept the language and makes a decision for every input string w. (TM can make decision for some input string though).
  A decision problem P called "undecidable" if the language L of all yes instances to P is not decidable. Undecidable languages are not recursive languages, but sometimes, they may be recursively enumerable languages



Example:

- The Halting Problem of Turing Machine
- The Mortality Problem
- The Mortal Matrix Problem.
- The Post Correspondance Problem, etc.

Q.6 State and Prove Rice's Theorem

Ans:

Rice Theorem

"Every property that is satisfied by some but not all recursively enumerable language is un-decidable."

- Any property that is satisfied by some recursively enumerable language but not all is known as non-trivial property.
- The Rice's theorem can be proved by reducing some other unsolvable problem to nontrivial property of recursively enumerable language

Theorem

$L = \{ \langle m \rangle \mid L(M) \in P \}$ is undecidable when P, a nontrivial property of the turing machine, is undecidable.

If the following two properties hold, it is proved as undecidable -

A. Property 1 - If m_1 and m_2 recognize the same language, then either $\langle m_1 \rangle \langle m_2 \rangle \in L$ or $\langle m_1 \rangle \langle m_2 \rangle \notin L$

B. Property 2 - For some m_1 and m_2 such that $\langle m_1 \rangle \in L$ and $\langle m_2 \rangle \notin L$

Proof -

Let there are two turing machines X1 and X2.

Let us assume $x1 \in L$ such that $L(X1) = \phi$ and $\langle x2 \rangle \notin L$.

For an input 'w' in a particular instant, perform:

→ If X accepts w, then simulate x2 on x.

→ Run Z on input $\langle w \rangle$

→ If Z accepts $\langle w \rangle$, reject it; and

If Z rejects $\langle w \rangle$, accept it.

If X accepts w, then
$L(w) = L(X2)$ and $\langle w \rangle \notin P$

If M does not accept w, then
$L(w) = L(X1) = \phi$ and $\langle w \rangle \in P$

Here the contradiction arises.
Hence, it is undecidable.