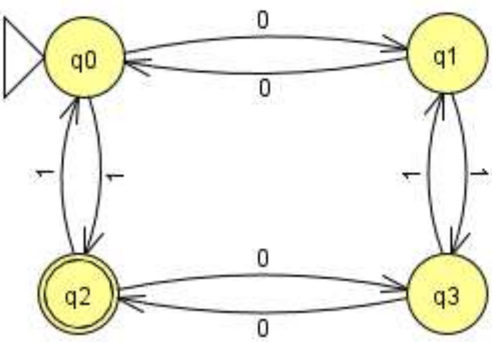
USN | | | | | | | | | |

**Internal Assessment Test 1 – Nov 2024**

| Sub: | Theory of Computation | | | | | Sub Code: | BCS503 | Branch: | CSE |
|---|---|---|---|---|---|---|---|---|---|
| Date: | 11.11.2024 | Duration: | 90 mins | Max Marks: | 50 | Sem/Sec: | 5 B | | OBE |

| | Answer any FIVE FULL Questions<br>**For all DFA, NFA questions, write the sample string in L and $\bar{L}$, the definition and transition table. Give Extended transition function for the strings give and state formally whether the string is accepted or rejected by the machine.* | MARKS | CO | RBT |
|---|---|---|---|---|
| 1 (a) | Define i)alphabet ii)Powers of alphabet with examples<br>An alphabet is a finite non-empty set of symbols. Σ denotes an alphabet.<br>1. Σ={0,1} is the binary alphabet<br>2. Σ={a,b,…,z} is the set of lower case letters<br>3. The set of all printable ASCII characters<br><br>ii) Powers of alphabet<br>If Σ is an alphabet, we can express the set of all strings of a certain length from that alphabet as $\Sigma^k$<br>$\Sigma^0 = \{\varepsilon\}$<br>Σ={0,1}, then $\Sigma^1 = \{0,1\}$, $\Sigma^2 = \{00,01,10,11\}$, $\Sigma^3 = \{000,001,010,011,100,101,110,111\}$<br>$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots$ | **3M** | **CO1** | **L1** |
| (b) | Design DFA for alphabet over {0,1}<br>i) Set of all strings where number of 0's is even and number of 1's is odd. Write extended transition function for $w_1$=001, $w_2$=1100<br><br><br>A= (Q,Σ,δ,$q_0$,F)<br>A=({ $q_0$, $q_1$, $q_2$, $q_3$}, {0,1}, δ, $q_0$, {$q_2$} ) | **7** | **CO1** | L3 |

| δ | 0 | 1 |
|---|---|---|
| → **$q_0$** | $q_1$ | $q_2$ |
| **$q_1$** | $q_0$ | $q_3$ |
| * **$q_2$** | $q_3$ | $q_0$ |
| **$q_3$** | $q_2$ | $q_1$ |

**Extended Transition Function**

i) $w_1$=001, $w_2$=1100

$$\hat{\delta}(q_0,\epsilon) = q_0$$

$$\hat{\delta}(q_0,0) = \delta(\hat{\delta}(q_0,\epsilon),0) = \delta(q_0,0) = q_1$$

$$\hat{\delta}(q_0,00) = \delta(\hat{\delta}(q_0,0),0) = \delta(q_1,0) = q_0$$

$$\hat{\delta}(q_0,001) = \delta(\hat{\delta}(q_0,00),1) = \delta(q_0,1) = q_2$$

**As $q_2 \in$ F, the string w=001 is accepted by the automaton A.**

$$\hat{\delta}(q_0,\epsilon) = q_0$$

$$\hat{\delta}(q_0,1) = \delta(\hat{\delta}(q_0,\epsilon),1) = \delta(q_0,1) = q_2$$

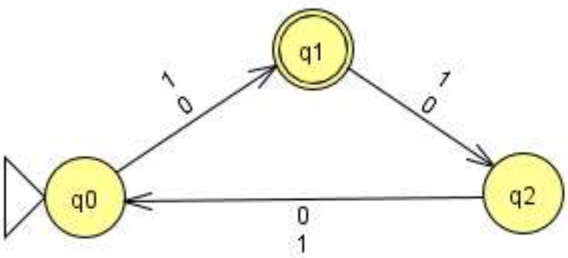
$$\hat{\delta}(q_0,11) = \delta(\hat{\delta}(q_0,1),1) = \delta(q_2,1) = q_0$$

$$\hat{\delta}(q_0,110) = \delta(\hat{\delta}(q_0,11),0) = \delta(q_0,0) = q_1$$

$$\hat{\delta}(q_0,1100) = \delta(\hat{\delta}(q_0,110),0) = \delta(q_1,0) = q_0$$

**As $q_0 \notin$ F, the string w=1100 is rejected by the automaton A.**

ii) {w: |w| mod 3 = 1} Write extended transition function for $w_1$=1001, $w_2$=010




A= (Q,Σ,δ,$q_0$,F)

A=({ $q_0$, $q_1$, $q_2$}, {0,1}, δ, $q_0$, {$q_1$} )

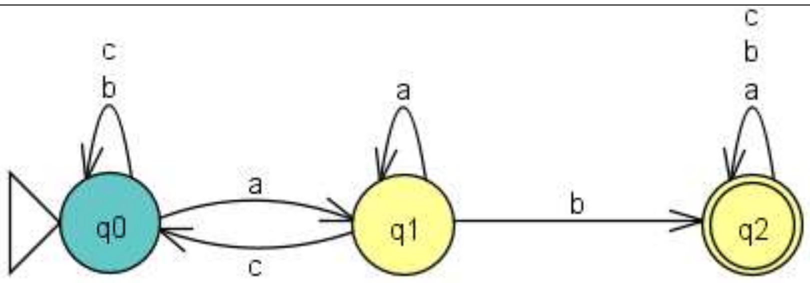
| δ | 0 | 1 |
|---|---|---|
| → $q_0$ | $q_1$ | $q_1$ |
| * $q_1$ | $q_2$ | $q_2$ |
| * $q_2$ | $q_0$ | $q_0$ |

**Extended Transition Function**

i) $w_1$=1001, $w_2$=010

$$\hat{\delta}(q_0,\epsilon) = q_0$$

| | | | | |
|---|---|---|---|---|
| | $\hat{\delta}(q_0,1) = \delta(\hat{\delta}(q_0,\epsilon),1) = \delta(q_0,1) = q_1$<br>$\hat{\delta}(q_0,10) = \delta(\hat{\delta}(q_0,1),0) = \delta(q_1,0) = q_2$<br>$\hat{\delta}(q_0,100) = \delta(\hat{\delta}(q_0,10),0) = \delta(q_2,0) = q_0$<br>$\hat{\delta}(q_0,1001) = \delta(\hat{\delta}(q_0,100),1) = \delta(q_0,1) = q_1$<br>**As q1 ∈F, the string w=1001 where \|w\| = 4 and 4 mod 3 = 1 is accepted by the automaton A.**<br>$\hat{\delta}(q_0,\epsilon) = q_0$<br>$\hat{\delta}(q_0,0) = \delta(\hat{\delta}(q_0,\epsilon),0) = \delta(q_0,0) = q_1$<br>$\hat{\delta}(q_0,01) = \delta(\hat{\delta}(q_0,0),1) = \delta(q_1,1) = q_2$<br>$\hat{\delta}(q_0,010) = \delta(\hat{\delta}(q_0,01),0) = \delta(q_2,0) = q_0$<br>**As q0 ∉ F, the string w=010 where \|w\| = 3 and 3 mod 3 = 0 is rejected by the automaton A.** | | | |
| 2 (a) | Define i)string ii)language<br>i) **String :** A string(word) is a finite sequence of symbols chosen from an alphabet. Eg. w=01101 is from the alphabet Σ={0,1}<br>Empty string is a string with 0 occurrences denoted by ε.<br>Length of a string : the number of positions for symbols in a string w=00101, \|w\| = 5.<br>Powers of alphabet can also be used to form strings from an alphabet of varying lengths.<br>ii) **Language :** A set of all strings which are chosen from $\Sigma^*$ where Σ is a particular alphabet.<br>If Σ is an alphabet, and $L \subseteq \Sigma^*$, then L is the language of Σ<br>Eg. 1. Set of all strings consisting of n 0's followed by n 1s, n ≥ 0<br>{ε,01,0011,000111,…<br>2. Set of binary numbers whose value is prime<br>{01,10,11,101,111,…} | **3** | **CO1** | L1 |
| (b) | Design DFA for alphabet over {a,b,c}<br>**i) Set of strings that have ab as a substring. Write extended transition function for $w_1$=cabc** | **7** | **CO1** | L3 |




L = {ab, cab, cabbc, aaccbab…}

$$\bar{L} = \{\varepsilon, aacc, bcaac, \dots\}$$

**Definition**

A= (Q,Σ,δ,$q_0$,F)

A=({ $q_0$, $q_1$, $q_2$}, {a,b,c}, δ, $q_0$, {$q_2$} )

| δ | a | b | c |
|---|---|---|---|
| → $q_0$ | $q_1$ | $q_0$ | $q_0$ |
| $q_1$ | $q_1$ | $q_2$ | $q_0$ |
| * $q_2$ | $q_2$ | $q_2$ | $q_2$ |

**Extended Transition Function**

$$\hat{\delta}\,(q_0, \epsilon) = q_0$$

$$\hat{\delta}\,(q_0, c) = \delta\big(\hat{\delta}\,(q_0, \epsilon), c\big) = \delta(q_0, c) = q_0$$

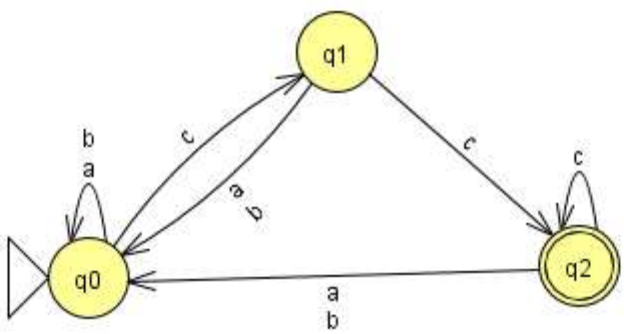
$$\hat{\delta}\,(q_0, ca) = \delta\big(\hat{\delta}\,(q_0, c), a\big) = \delta(q_0, a) = q_1$$

$$\hat{\delta}\,(q_0, cab) = \delta\big(\hat{\delta}\,(q_0, ca), b\big) = \delta(q_1, b) = q_2$$

$$\hat{\delta}\,(q_0, cabc) = \delta\big(\hat{\delta}\,(q_0, cab), c\big) = \delta(q_2, c) = q_2$$

As $q_2$ ∈F, the string w=cabc is accepted by the automaton A.

**ii) Set of strings that end in cc. Write extended transition function for $w_1$=abcc**




L = {cc, abcc, bacacc…}

$$\bar{L} = \{\varepsilon, a, b, c, aa, ab, baccab, \ldots\}$$

**Definition**

A= (Q,Σ,δ,$q_0$,F)

A=({ $q_0$, $q_1$, $q_2$}, {a,b,c}, δ, $q_0$, {$q_2$} }

| δ | a | b | c |
|---|---|---|---|
| **→ $q_0$** | $q_0$ | $q_0$ | $q_1$ |
| **$q_1$** | $q_0$ | $q_0$ | $q_2$ |
| *** $q_2$** | $q_0$ | $q_0$ | $q_2$ |

**Extended Transition Function**

$$\hat{\delta}\,(q_0, \epsilon) = q_0$$

$$\hat{\delta}\,(q_0, a) = \delta(\hat{\delta}\,(q_0, \epsilon), a) = \delta(q_0, a) = q_0$$

$$\hat{\delta}\,(q_0, ab) = \delta(\hat{\delta}\,(q_0, a), b) = \delta(q_0, b) = q_0$$

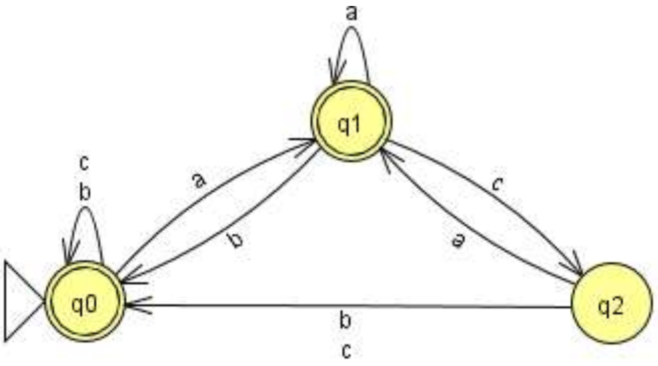
$$\hat{\delta}\,(q_0, abc) = \delta(\hat{\delta}\,(q_0, ab), c) = \delta(q_0, c) = q_1$$

$$\hat{\delta}\,(q_0, abcc) = \delta(\hat{\delta}\,(q_0, abc), c) = \delta(q_1, c) = q_2$$

As $q_2$ ∈F, the string w=abcc is accepted by the automaton A.

**iii) Set of strings that do not end in ac. Write extended transition function for $w_1$=cca**

First design a machine that ends in ac, then invert the accepting and non-accepting states




L = {ab, cabb, aaca}

$$\bar{L} = \{ac, abac, aaac, \ldots\}$$

**Definition**

A= (Q,Σ,δ,$q_0$,F)

A=({ $q_0$, $q_1$, $q_2$}, {a,b,c}, δ, $q_0$, {$q_0$, $q_1$} }

| δ | a | b | c |
|---|---|---|---|
| ***→ $q_0$** | $q_1$ | $q_0$ | $q_0$ |
| ***$q_1$** | $q_1$ | $q_0$ | $q_2$ |
| **$q_2$** | $q_1$ | $q_0$ | $q_0$ |

**Extended Transition Function**

$$\hat{\delta}\,(q_0, \epsilon) = q_0$$

$$\hat{\delta}\,(q_0, c) = \delta\big(\hat{\delta}\,(q_0, \epsilon), c\big) = \delta(q_0, c) = q_0$$

$$\hat{\delta}\,(q_0, cc) = \delta\big(\hat{\delta}\,(q_0, c), c\big) = \delta(q_0, c) = q_0$$

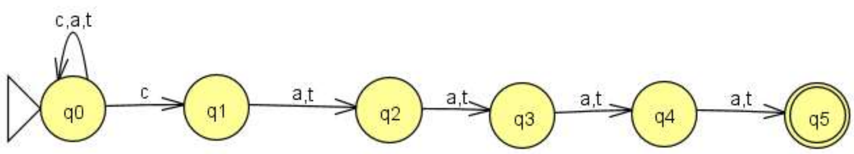
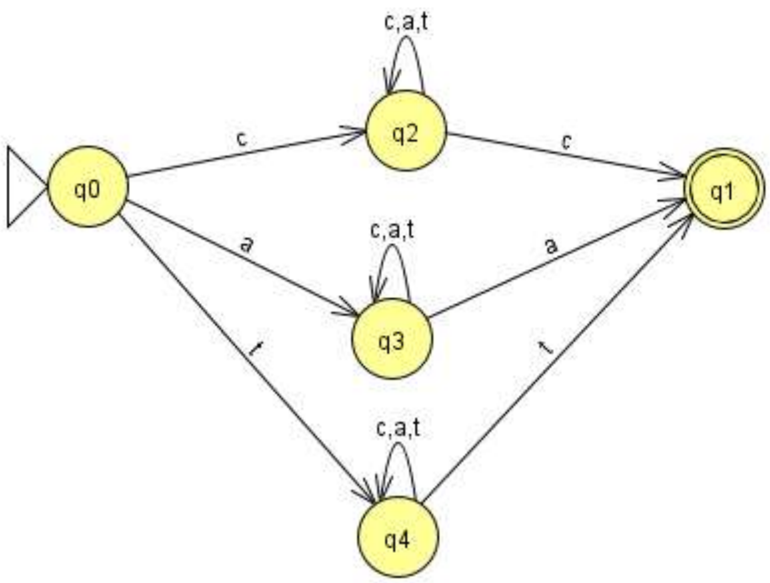
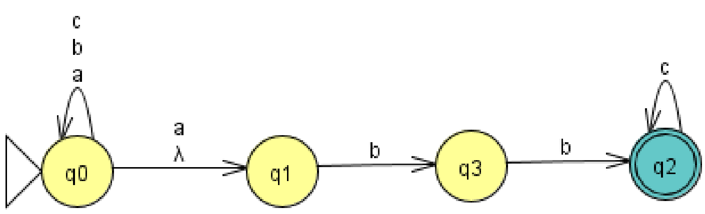
$$\hat{\delta}\,(q_0, cca) = \delta\big(\hat{\delta}\,(q_0, cc), a\big) = \delta(q_0, a) = q_1$$

As $q_1$ ∈F, the string w=cca is accepted by the automaton A.

3 (a) | Marks: 4 | **CO1** | L1

Differentiate NFA and DFA. (Clearly explain the difference in transition functions formally)

| DFA | NFA |
|---|---|
| Deterministic | Non-deterministic |
| Transition function for each input symbol to exactly one state<br>δ(s,a) = r where r∈Q | Transition relation on input symbol results in a Φ set or more than one states<br>Eg. δ(s,a) = Φ<br>Δ(s,b) = {p,r} |
| Running time is O(n) where n is the length of the input string | Worst case time complexity is O(N*S) where N is the length of the strings and S is the number of states. |
| There is exactly one path to accept or reject a string | There are multiple paths. Some paths may lead to accepting states. |
| A string is accepted when<br>$\hat{\delta}\,(q_0, w) = f \ and\ f \in F$ | A string is accepted when<br>$\hat{\delta}\,(q_0, w) = \{a_1, a_2, \ldots a_n\}$ where $= \{a_1, a_2, \ldots a_n\} \cap F \neq \Phi$ |
| Difficult to construct | Simple to construct |

<table>
<tr><td></td><td><table><tr><td>Does not contain ε-Transitions</td><td>May contain ε- transitions</td></tr></table></td><td></td><td></td><td></td></tr>
<tr><td>(b)</td><td>Design NFA for the following over the alphabet {c,a,t}<br>i) Accepts strings where the 5th last letter from the right is c, eg. taacaatc<br>
<br>ii) Accepts strings that begins and ends with the same letter.<br>
</td><td>6</td><td>CO1</td><td>L3</td></tr>
<tr><td>4 (a)</td><td>Write the procedure to compute ε-CLOSE<br>Basis :The state q is in ε-CLOSE(q)<br>Induction: If state p is is in ε-CLOSE(q) and there is a transition from state p to state r labeled ε, then r is in ε-CLOSE(q)</td><td>3</td><td>CO1</td><td>L1</td></tr>
<tr><td>(b)</td><td>Convert the following ε-NFA (λ is empty transition) to a DFA by first computing ε-CLOSE of all the states.<br>
<br><table><tr><th>δ</th><th>a</th><th>b</th><th>c</th><th>ε-close</th></tr><tr><td>→ $q_0$</td><td>{$q_0$, $q_1$}</td><td>$q_0$</td><td>$q_0$</td><td>{$q_0$,$q_1$}</td></tr><tr><td>$q_1$</td><td>Φ</td><td>$q_3$</td><td>Φ</td><td>{ $q_1$}</td></tr></table></td><td>7</td><td>CO1</td><td>L3</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| **$q_3$** | Φ | $q_2$ | Φ | $\{q_3\}$ |
| **$*q_2$** | Φ | Φ | $q_2$ | $\{q_2\}$ |

**<u>Lazy Subset Construction</u>**

$\delta(\{q_0,q_1\}, a) = \delta(q_0,a) \cup \delta(q_1,a) = \{q_0, q_1\} \cup \Phi = \varepsilon\text{-close}(q_0) \cup \varepsilon\text{-close}(q_1) = \{q_0, q_1\} \cup \{q_1\} = \{q_0, q_1\}$

$\delta(\{q_0,q_1\}, b) = \delta(q_0,b) \cup \delta(q_1,b) = \{q_0\} \cup \{q_3\} = \varepsilon\text{-close}(q_0) \cup \varepsilon\text{-close}(q_3) = \{q_0, q_1\} \cup \{q_3\} = \{q_0, q_1, q_3\}$

$\delta(\{q_0,q_1\}, a) = \delta(q_0,c) \cup \delta(q_1,c) = \{q_0\} \cup \Phi = \varepsilon\text{-close}(q_0) = \{q_0, q_1\}$

$\delta(\{q_0,q_1, q_3\}, a) = \delta(q_0,a) \cup \delta(q_1,a) \cup \delta(q_3,a) = \{q_0, q_1\} \cup \Phi \cup \Phi = \varepsilon\text{-close}(q_0) = \{q_0, q_1\}$

$\delta(\{q_0,q_1, q_3\}, b) = \delta(q_0,b) \cup \delta(q_1,b) \cup \delta(q_3,b) = \{q_0\} \cup \{q_3\} \cup \{q_2\} = \varepsilon\text{-close}(q_0) \cup \varepsilon\text{-close}(q_3) \cup \varepsilon\text{-close}(q_2) = \{q_0, q_1\} \cup \{q_3\} \cup \{q_2\} = \{q_0, q_1, q_3. q_2\}$

$\delta(\{q_0,q_1, q_3\}, c) = \delta(q_0,c) \cup \delta(q_1,c) \cup \delta(q_3,c) = \{q_0\} \cup \Phi \cup \Phi = \varepsilon\text{-close}(q_0) = \{q_0, q_1\}$

$\delta(\{q_0,q_1, q_3, q_2\}, a) = \delta(q_0,a) \cup \delta(q_1,a) \cup \delta(q_3,a) \cup \delta(q_2,a) = \{q_0, q_1\} \cup \Phi \cup \Phi \cup \Phi = \varepsilon\text{-close}(q_0) = \{q_0, q_1\}$

$\delta(\{q_0,q_1, q_3, q_2\}, b) = \delta(q_0,b) \cup \delta(q_1,b) \cup \delta(q_3,b) \cup \delta(q_2,a) = \{q_0\} \cup \{q_3\} \cup \{q_2\} \cup \Phi = \varepsilon\text{-close}(q_0) \cup \varepsilon\text{-close}(q_3) \cup \varepsilon\text{-close}(q_2) = \{q_0, q_1\} \cup \{q_3\} \cup \{q_2\} = \{q_0, q_1, q_3. q_2\}$

$\delta(\{q_0,q_1, q_3, q_2\}, c) = \delta(q_0,c) \cup \delta(q_1,c) \cup \delta(q_3,c) \cup \delta(q_2,c) = \{q_0\} \cup \Phi \cup \Phi \cup \{q_2\} = \varepsilon\text{-close}(q_0) \cup \varepsilon\text{-close}(q_2) = \{q_0, q_1\} \cup \{q_2\} = \{q_0, q_1, q_2\}$

$\delta(\{q_0,q_1, q_2\}, a) = \delta(q_0,a) \cup \delta(q_1,a) \cup \delta(q_2,a) = \{q_0, q_1\} \cup \Phi \cup \Phi = \varepsilon\text{-close}(q_0) = \{q_0, q_1\}$

$\delta(\{q_0,q_1, q_2\}, b) = \delta(q_0,b) \cup \delta(q_1,b) \cup \delta(q_2,b) = \{q_0\} \cup \{q_3\} \cup \Phi = \varepsilon\text{-close}(q_0) \cup \varepsilon\text{-close}(q_3) = \{q_0, q_1\} \cup \{q_3\} = \{q_0, q_1, q_3\}$

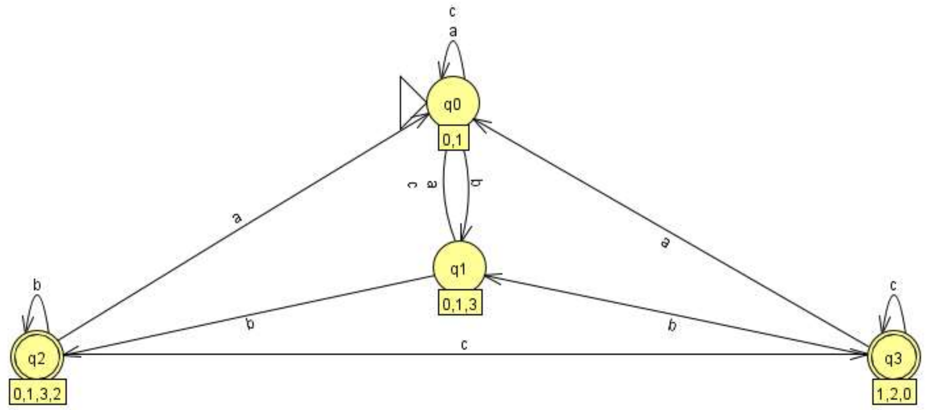
$\delta(\{q_0,q_1,q_2\}, c) = \delta(q_0,c) \cup \delta(q_1,c) \cup \delta(q_2,c) = \{q_0\} \cup \Phi \cup \{q_2\} = \varepsilon\text{-close}(q_0)$

$\cup\ \varepsilon\text{-close}(q_2) = \{q_0, q_1\} \cup \{q_2\} = \{q_0, q_1, q_2\}$

| $\delta_D$ | a | b | c |
|---|---|---|---|
| **$\rightarrow \{q_0,q_1\}$** | $\{q_0, q_1\}$ | $\{q_0, q_1, q_3\}$ | $\{q_0, q_1\}$ |
| **$\{q_0,q_1,,q_3\}$** | $\{q_0, q_1\}$ | $\{q_0, q_1, q_3. q_2\}$ | $\{q_0, q_1\}$ |
| **$*\{q_0, q_1, q_3. q_2\}$** | $\{q_0, q_1\}$ | $\{q_0, q_1, q_3. q_2\}$ | $\{q_0, q_1, q_2\}$ |
| **$*\{q_0, q_1, q_2\}$** | $\{q_0, q_1\}$ | $\{q_0, q_1, q_3\}$ | $\{q_0, q_1, q_2\}$ |

$A = (Q,\Sigma,\delta,q_0,F)$

$A=(\{ \{q_0,q_1\},\{q_0,q_1\},\{q_0, q_1, q_3. q_2\},\{q_0, q_1, q_2\} \}, \{a,b,c\}, \delta_D, \{q_0,q_1\}, \{\{q_0, q_1, q_3. q_2\}, \{q_0, q_1, q_2\}\} )$

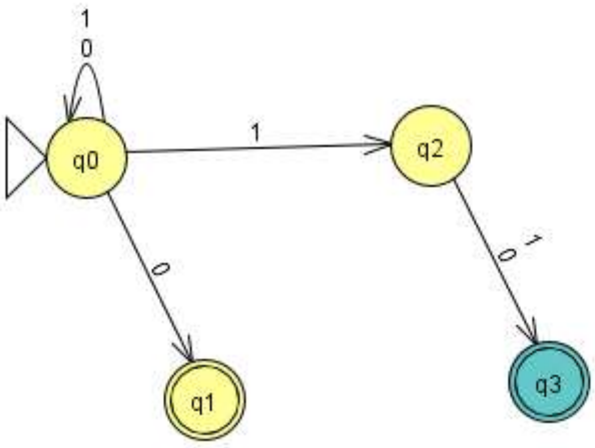



**5 (a)** Define NFA [4 | CO1 | L1]

$A = (Q,\Sigma,\delta,q_0,F)$

1. Q is the finite set of states
2. $\Sigma$ is the finite set of input symbols
3. $q_0 \in Q$ is the start state
4. $\delta$ is the transition function that takes a state in Q and an input symbol in $\Sigma$ and returns a subset of Q. If there is no state, it returns $\Phi$.

**(b)** Convert the following NFA into DFA using subset construction. Expand only accessible states. Write the Definition and transition table of the DFA. [6 | CO1 | L3]




| δ | 0 | 1 |
|---|---|---|
| **→ $q_0$** | $\{q_0, q_1\}$ | $\{q_0, q_2\}$ |
| **$q_1$** | Φ | Φ |
| **$q_2$** | $\{q_3\}$ | $\{q_3\}$ |
| ***$q_3$** | Φ | Φ |

**Lazy Subset Construction**

$\delta(\{q_0\}, 0) = \{q_0, q_1\}$ , $\delta(\{q_0\}, 1) = \{q_0, q_2\}$

$\delta(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \Phi = \{q_0, q_1\}$

$\delta(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0, q_2\} \cup \Phi = \{q_0, q_2\}$

$\delta(\{q_0, q_2\}, 0) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \{q_3\} = \{q_0, q_1, q_3\}$

$\delta(\{q_0, q_2\}, 1) = \delta(q_0, 1) \cup \delta(q_2, 1) = \{q_0, q_2\} \cup \{q_3\} = \{q_0, q_2, q_3\}$

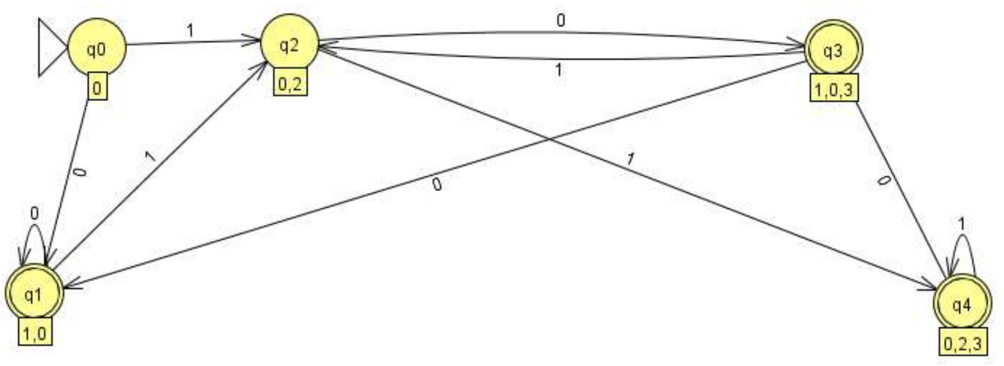
$\delta(\{q_0, q_1, q_3\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_3, 0) = \{q_0, q_1\} \cup \Phi \cup \Phi = \{q_0, q_1\}$

$\delta(\{q_0, q_1, q_3\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_3, 1) = \{q_0, q_2\} \cup \Phi \cup \Phi = \{q_0, q_2\}$

$\delta(\{q_0, q_2, q_3\}, 0) = \delta(q_0, 0) \cup \delta(q_2, 0) \cup \delta(q_3, 0) = \{q_0, q_1\} \cup \{q_3\} \cup \Phi = \{q_0, q_1, q_3\}$

$\delta(\{q_0, q_2, q_3\}, 1) = \delta(q_0, 1) \cup \delta(q_2, 1) \cup \delta(q_3, 1) = \{q_0, q_2\} \cup \{q_3\} \cup \Phi = \{q_0, q_2, q_3\}$

| $\delta_D$ | 0 | 1 |
|---|---|---|
| $\rightarrow$ **$q_0$** | $\{q_0, q_1\}$ | $\{q_0, q_2\}$ |
| ***$\{q_0, q_1\}$** | $\{q_0, q_1\}$ | $\{q_0, q_2\}$ |
| **$\{q_0, q_2\}$** | $\{q_0, q_1, q_3\}$ | $\{q_0, q_2, q_3\}$ |
| ***$\{q_0, q_1, q_3\}$** | $\{q_0, q_1\}$ | $\{q_0, q_2\}$ |
| ***$\{q_0, q_2, q_3\}$** | $\{q_0, q_1, q_3\}$ | $\{q_0, q_2, q_3\}$ |




$A = (Q, \Sigma, \delta, q_0, F)$

$A = (\{\{q_0\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_0, q_1, q_3\}, \{q_0, q_2, q_3\}\}, \{0,1\}, \delta_D, \{q_0\}, \{\{q_0, q_1\}, \{q_0, q_1, q_3\}, \{q_0, q_2, q_3\}\})$

---

6 (a)

What are Regular expressions? Explain the construction of regular expression.

A regular expression is a declarative way to express strings we want to accept.

Building RE

**Basis :**

1. ε and Φ are regular expressions denoting the languages {ε} and Φ
2. If a is a symbol, then a is a regular expression {a} and L(a) = {a}
3. A variable L representing the language

**Induction:**

1. If E and F are regular expressions, the union of E and F is also a RE
   E+F, L(E+F) = L(E) + L(F).
2. If E and F are regular expressions, the concatenation of E and F is also a RE
   EF, L(EF) = L(E) L(F)
3. If E is a regular expression E* is also RE denoting closure of E. L(E*) =(L(E))*

**4** | **CO2** | L2

| | | | | |
|---|---|---|---|---|
| | 4. If E is a regular expression, (E) is also RE. | | | |
| (b) | Form regular expressions for the following:<br>For alphabet over {0,1}<br>i) The length of the string is odd<br>(0+1) ( (0+1)(0+1) )*<br>ii) Contains the substring 001<br>(0+1)* 001 (0+1)*<br>iii) Number of 0s is a multiple of 3<br>(1*01*01*01*)*<br>For alphabet over {t,o,c}<br>iv) Containing at least 2 ***o***'s<br>(t+c+o)* o (t+c+o)* o (t+c+o)*<br>v) $t^n c^m o^n$: n>=2, m<3<br>ttt*(ε + c + cc)ooo*<br>vi) at least one ***c*** and at least one ***t***<br>(t+c+o)*c(t+c+o)*t (t+c+o)* +<br>(t+c+o)*t(t+c+o)*c(t+c+o)* | **6** | **CO2** | L3 |

**CI** **CCI** **HOD**

| Course Outcomes | | Blooms Level | Modules covered | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 | PSO3 | PSO4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CO1 | Apply the fundamentals of automata theory to write DFA, NFA, Epsilon-NFA and conversion between them. | **L1, L2, L3** | **1** | 3 | 2 | 2 | - | 2 | - | - | - | - | - | - | - | 2 | 2 | - | 3 |
| CO2 | Prove the properties of regular languages using regular expressions. | **L1, L2** | **2** | 3 | 3 | 2 | 3 | - | - | - | - | - | - | - | - | 2 | 2 | - | 3 |
| CO3 | Design context-free grammars (CFGs) and pushdown automata (PDAs) for formal languages. | **L1, L2, L3** | **3,4** | 3 | 3 | 2 | 3 | 2 | - | - | - | - | - | - | - | - | 2 | - | 3 |
| CO4 | Design Turing machines to solve the computational problems. | **L1, L2, L3** | **5** | 2 | 3 | 2 | 3 | 2 | - | - | - | - | - | - | - | - | 2 | - | 3 |
| CO5 | Explain the concepts of decidability and undecidability | **L1, L2, L3** | **5** | 3 | 2 | 2 | 3 | - | - | - | - | - | - | - | - | - | 2 | - | 3 |

**CO PO Mapping**

| COGNITIVE LEVEL | REVISED BLOOMS TAXONOMY KEYWORDS |
|---|---|
| L1 | List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc. |
| L2 | summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend |
| L3 | Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover. |
| L4 | Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer. |
| L5 | Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize. |

| PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO) | | | | CORRELATION LEVELS | |
|---|---|---|---|---|---|
| PO1 | Engineering knowledge | PO7 | Environment and sustainability | 0 | No Correlation |
| PO2 | Problem analysis | PO8 | Ethics | 1 | Slight/Low |
| PO3 | Design/development of solutions | PO9 | Individual and team work | 2 | Moderate/ Medium |

| | | | | | |
|---|---|---|---|---|---|
| PO4 | Conduct investigations of complex problems | PO10 | Communication | 3 | Substantial/ High |
| PO5 | Modern tool usage | PO11 | Project management and finance | | |
| PO6 | The Engineer and society | PO12 | Life-long learning | | |
| PSO1 | Develop applications using different stacks of web and programming technologies | | | | |
| PSO2 | Design and develop secure, parallel, distributed, networked, and digital systems | | | | |
| PSO* 3 | Apply software engineering methods to design, develop, test and manage software systems. | | | | |
| PSO4 | Develop intelligent applications for business and industry | | | | |