1901CS63

CS303 MIDSEM TANISHQ MALU

Q1

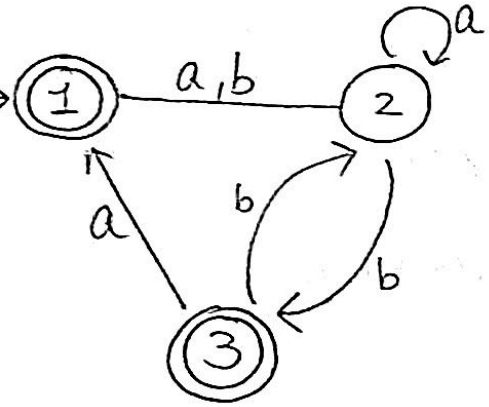
(a) $\Sigma = \{0, 1\}$, L = even no. of 0s ~~and~~ or exactly two 1s

Sol $1^*(01^*01^*) + (0^*10^*10^*)$

(b) $L = \{\varepsilon, 0\}$

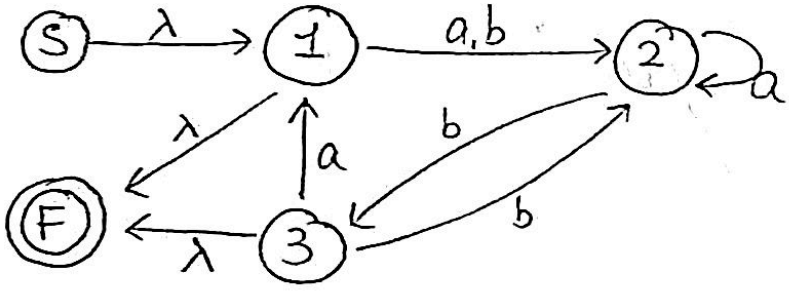
Sol $0 + \varepsilon$

(c) DFA ⇒



Converting our DFA to GNFA to find the regular expression.

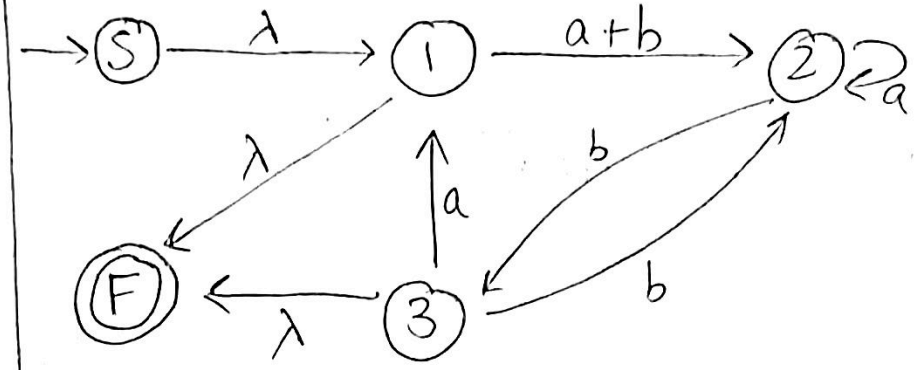
1) Adding new start state and new final state. Making original final states as non final. Adding required ε transition

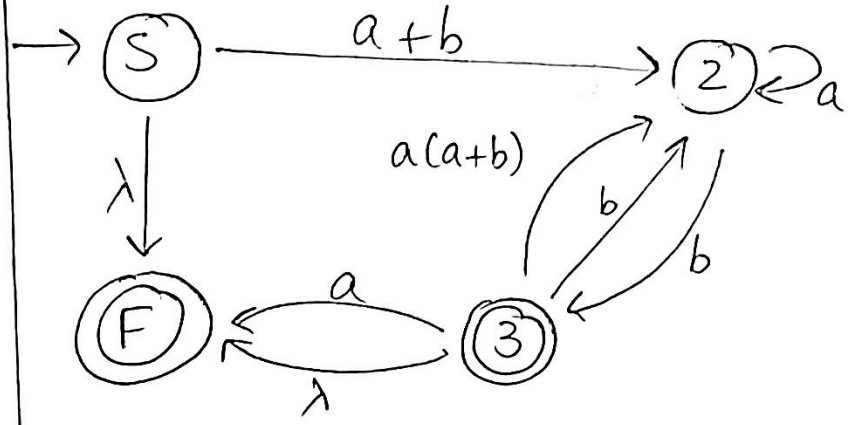


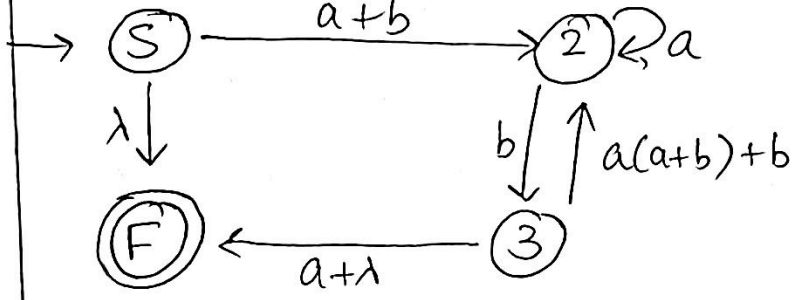
2) Reducing states ⇒ (1.) Union : $A \xrightarrow{0} B \Rightarrow A \xrightarrow{0+1} B$

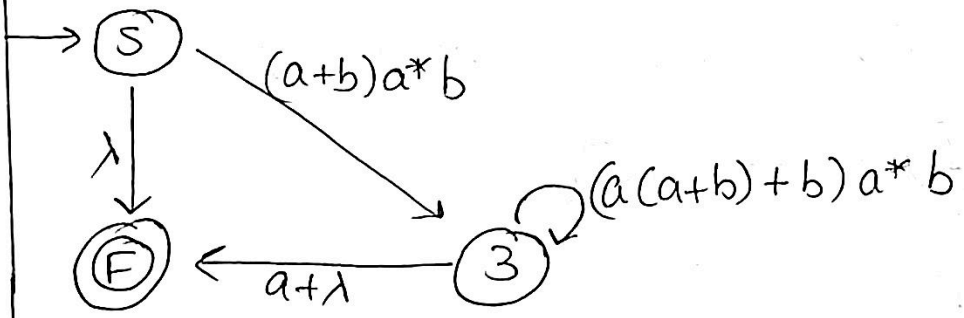
(2.) Internal state : $A \xrightarrow{x} R \xrightarrow{y} B \Rightarrow A \xrightarrow{xy} B$

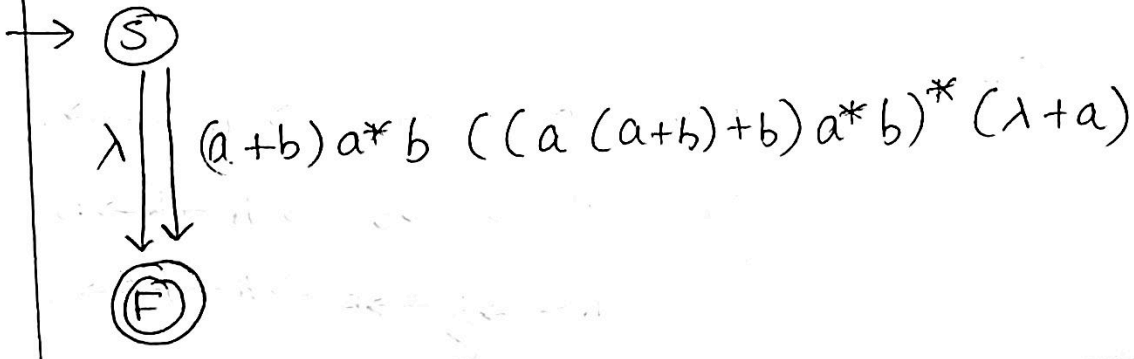
$A \xrightarrow{x} R \xrightarrow{z} B \Rightarrow A \xrightarrow{xy^*z} B$ (with y looping on R)


S
λ
1
a+b
2
a
λ
a
b
F
λ
3
b


S
a+b
2
a
λ
a(a+b)
b
b
F
a
λ
3


S
a+b
2
a
λ
b
a(a+b)+b
F
a+λ
3


S
(a+b)a*b
λ
(a(a+b)+b)a*b
F
a+λ
3


S
λ
(a+b)a*b((a(a+b)+b)a*b)*(λ+a)
F

Final regex

$\Rightarrow \lambda + ((a+b)a^*b)((a(a+b)+b)a^*b)^*(\lambda+a)$

(d) $\Sigma = \{a, b\}$ L= odd no. of as and ends with ab

Sol $b^*(ab^*ab^*)^*ab$

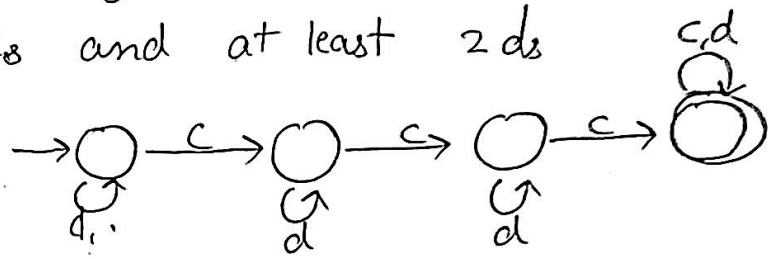
(e) $\Sigma = \{0, 1\}$ L= strings except 11 and 111

Sol $(\varepsilon+1) + (0 + 10 + 110 + 1110 + 1111)(0+1)^*$

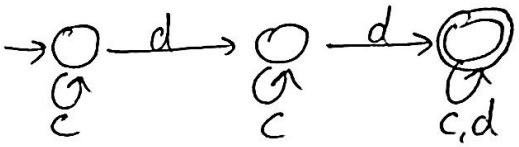
Q2 Draw DFA for $\Sigma = \{c, d\}$

(a) w|w has at least 3 cs and at least 2 ds

Sol DFA for at least 3 cs

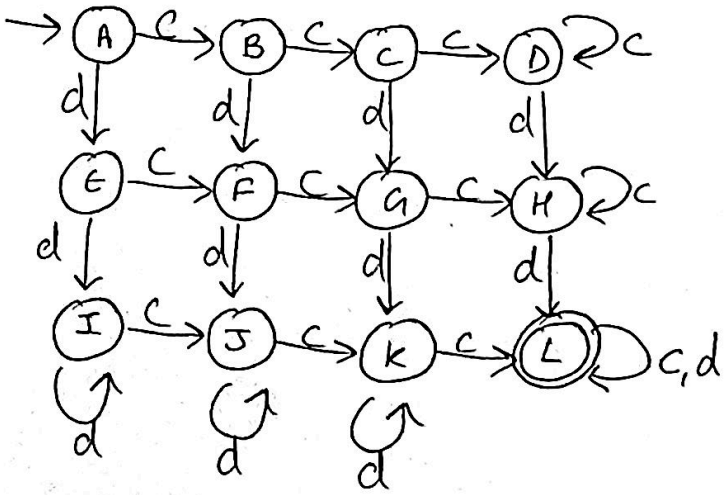


DFA for at least 2 ds



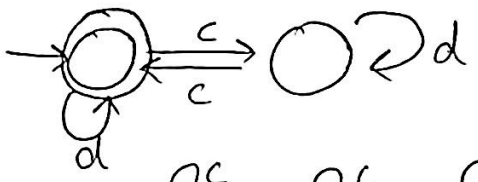
On taking intersection

DFA =>



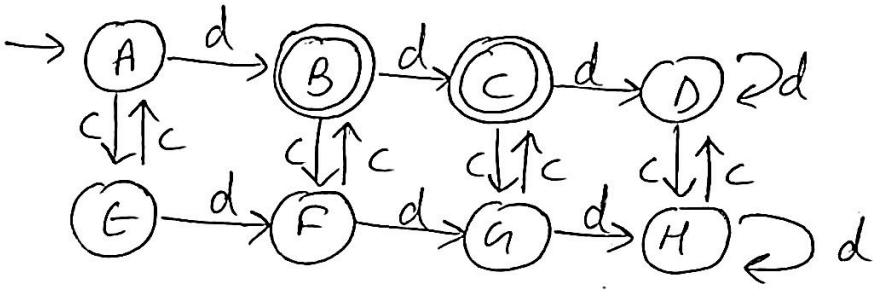
(b) w | w has even no. of cs and one or two ds

DFA for even cs



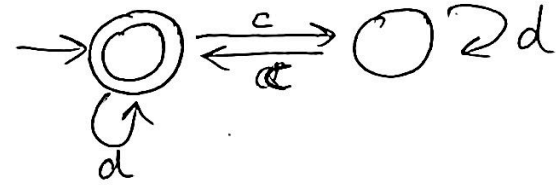
DFA for one or two d

Intersection ⇒

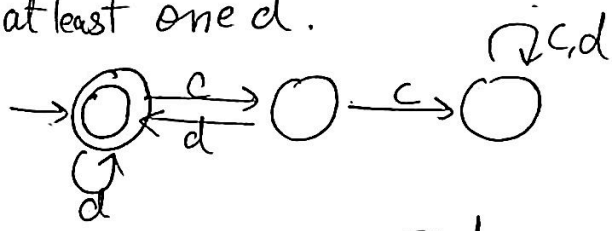


(c) w | w has an even no. of cs and each c is followed by at least one d.

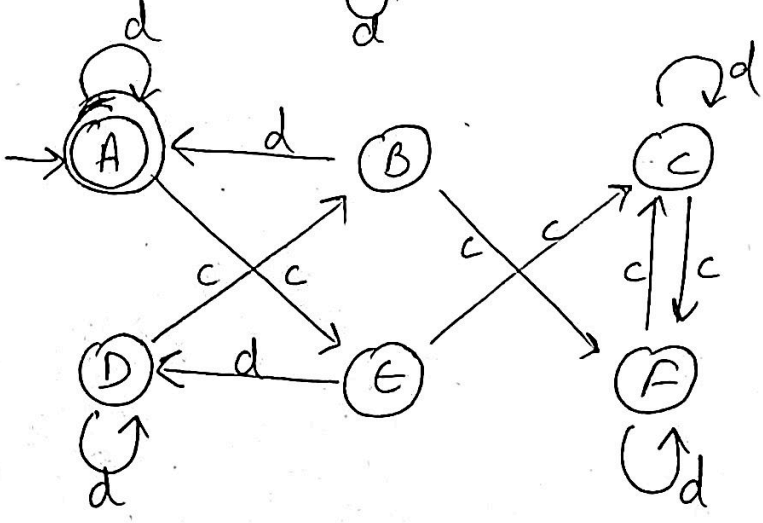
DFA for even cs -



DFA for each c followed by at least one d.

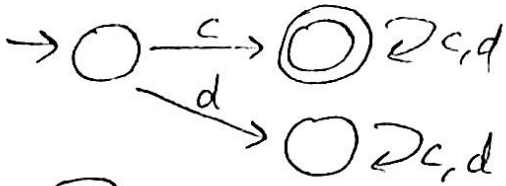


Intersection ⇒

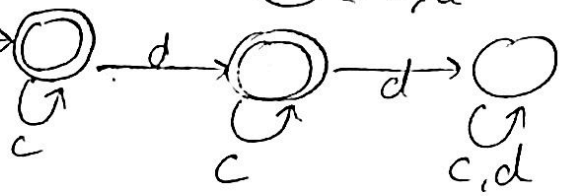


(d) w|w start with an c and has at most one d.

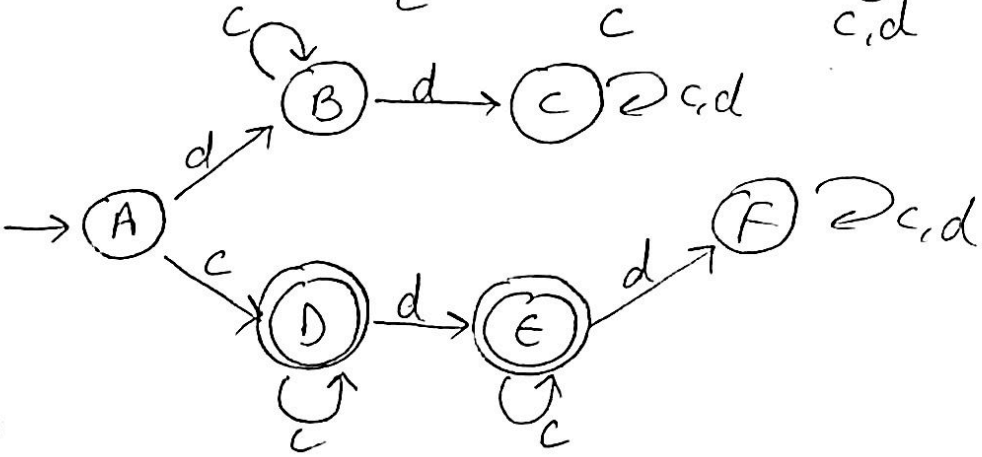
- DFA for start with c



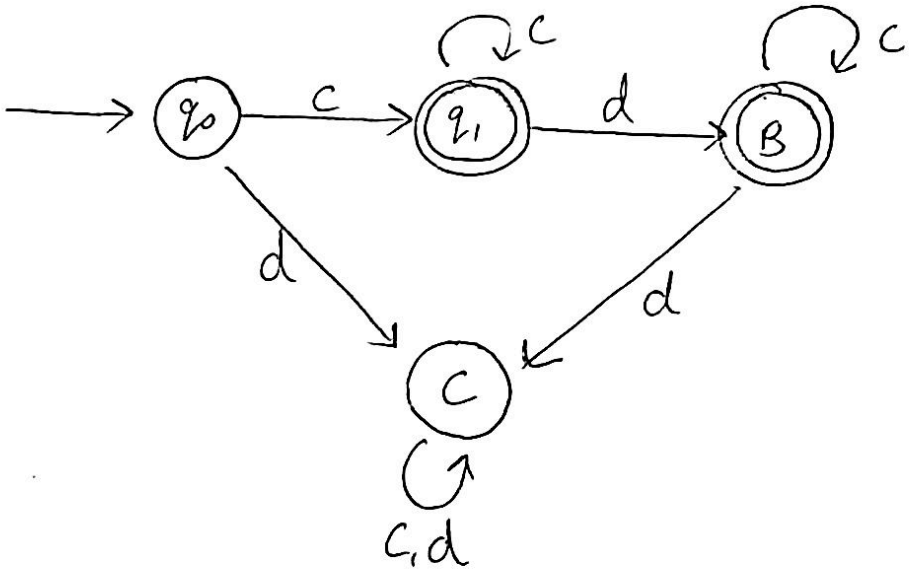
- DFA for at most one d



Intersection:



Reduced DFA



Q3

Q1 Given a regular language A over Σ, let

Delete(A) = $\{ xz \mid x, z \in \Sigma^* \text{ and } xyz \in A \text{ for some } y \in \Sigma \}$

To show that if A is regular then delete(A) is regular.

⇒ Let A be a regular language, and D a DFA that recognizes it. Lets construct an NFA N than recognizes Delete(A) (let it be L(N)).

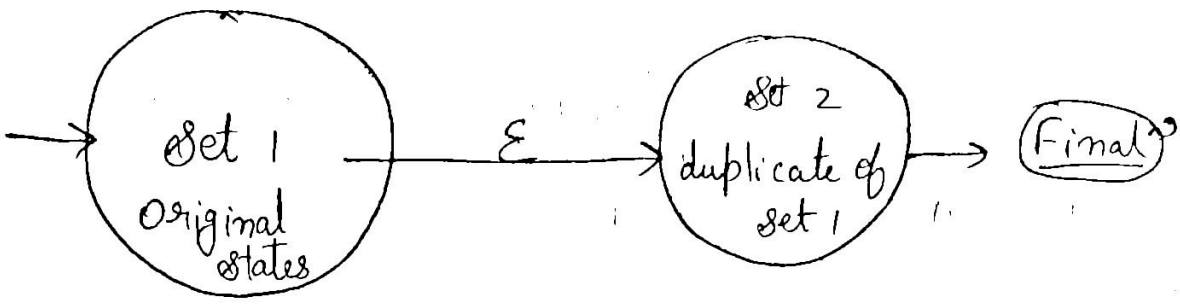
⇒ The basic idea of our NFA N is to have a second set of states, which are exactly similar to DFA, D. Also it has an option to choose a λ transition, effectively guessing as to what the deleted character might. while there will be no λ transition in second set of states indicating that one character has already been deleted.

⇒ Let $D = (Q, \Sigma, \delta, q_0, F)$

$N = (Q \cup Q', \Sigma, \delta', q_0, F')$

where $Q' = \{ q' \mid \text{where } \forall\, q \in Q \ (q)' = q' \}$

$F' = \{ q' \mid \text{where } \forall\, q \in F \ (q)' = q' \}$



Schematic view of NFA

→ δ' is given by $\delta'(q, c) = \{\delta(q, c)\}$

$\delta'(q', c) = (\delta(q, c))'$

$\delta(q, \varepsilon) = \{(\delta(q, a))' \mid a \in \Sigma\}$

$\delta'(q', \varepsilon) = \phi$

We have to show that $L(N) = \text{Delete}(A)$

(i) $L(N) \subseteq \text{Delete}(A)$

Let $s \in L(N)$, then by definition we know that, $s = s_1 s_2 s_3 \ldots s_n$ where $s_i \in \Sigma_\varepsilon$ and we have $t_0 t_1 \ldots t_n$ where $t_0 = q_0$ = start state of DFA-D and $t_n \in F'$ and each $t_{i+1} \in \delta'(t_i, s_{i+1})$. Now, $t_n \in F' \subseteq Q'$, lets choose the smallest k such that $t_k \in Q'$. Thus $t_i \in Q'$ for all $i \geq k$ since there is no transition from set 2 to set 1 Q. Now following inferences can be made :—

(i) $s_i \neq \varepsilon$ for $i > k$ since no $\delta'(t_j, s_{j+1})$ can be empty of $0 \leq j < n$

(ii) If $q \in Q$ and $\delta(q, c) \in Q'$ then $c = \varepsilon$ (by definition of NFA)

(iii) $s_i \neq \varepsilon$ for $i < k$ (by the minimality of k)

$\therefore t_{k-1} \in Q$ and $t_k \in Q'$

So this means $\boxed{s_k = \varepsilon}$

$\Rightarrow$ Now,

For some $q_{k-1} \in \delta'(q_{k-1}, \varepsilon)$, keeping in mind δ', there must be some $a \in \Sigma$ such ~~that for $0 \leq i < k-1$. similarly for $i \geq k$.~~ that $\delta(q_{k-1}, a) = q_k$. Further we have $t_{i+1} \in \delta'(t_i, s_i) = \{\delta(t_i, s_i)\}$ and thus $t_{i+1} = \delta(t_i, s_i)$ for $0 \leq i < k-1$. similarly for $i \geq k$ we have $t_i \in Q'$ say $(t_i) = (t_i)'$ and then $'t_{i+1} = \delta('t_i, s_{i+1})$

Let $s' = s_0 \ldots s_{k-1}\, a\, s_k \ldots s_n \in \Sigma^*$ and consider $t_0 \ldots t'_{k-1}\, t_k \ldots {}'t_n$. we have every $t_i \in Q$, $t_0 = q_0$, ${}'t_n \in F$ and

$$\delta(t_i, s_{i+1}) = t_{i+1} \quad 0 \leq i < k-1$$

$$\delta(t_{k-1}, a) = t_k$$

$$\delta('t_i, s_{i+1}) = {}'t_{i+1} \quad k \leq i < n$$

Therefore, by definition, D accepts s', so $s' \in L$

Further we have $s \in$ Delete (A) with $s = xyz$

$x = s_0 s_1 \cdots s_{k-1}$ $\quad y = a \quad$ $z = s_{k+1} \cdots s_n$

Thus N accepts only strings in Delete (A).

$L(N) \supseteq$ Delete (A)

Let $s \in$ Delete (A) and let x, y, z be such that $x, z \in \Sigma^*$ and $y \in \Sigma$ and $xz = s$ and $s' = xyz \in A$. Then D accepts xyz, so we have $s' = s'_0 \cdots s'_n$ and a sequence of states $t_0 \ldots t_n$ where $t_n \in F$ and $\delta(t_i, s'_{i+1}) = t_{i+1}$ for $0 \le i < n$. Let $k = |x|$, (so s_k would be y) and $s = s'_0 \cdots s'_{k-1} \, s'_{k+1} \cdots s'_n$ and let $q_0 \ldots q_n = t_0 \cdots t_{k-1} \, t'_k \, t'_{k+1} \ldots t'_n$. we have $q_0 =$ start state of Q and $q_n \in F'$ so q_n = finish state of F'. Now for each $0 \le i < k-1$

$q_{i+1} = t_{i+1} \in \{t_{i+1}\} = \delta'(t_i, s_i)$;

for $i = k-1$

$$q_{i+1} = t'_k \in \{\delta(t_i, a) \mid a \in \Sigma\} = \delta'(t_i, \Sigma)$$

$$= \delta'(q_i, s_{i+1})$$

Now since

$$\delta(t_i, y) = t_k$$

and for $i \leq k < n$

$$q_{i+1} = t'_{i+1} \in \{t'_{i+1}\} = \delta'(t'_i, \sigma_i) = \delta'(q_i, \sigma_i)$$

Thus by definition, N accepts σ.

Therefore N accepts every string of Delete(A).

Hence Delete(A) is regular if A is regular.

Hence proved.

Q4

(a) Consider a language of all binary strings with twice as many 0s as 1s.

Sol Let us consider a CFG $G_1 = (V, T, S, P)$ which can generate the required language. Therefore, this

$G_1 = \{\{0,1,S\}, \{0,1\}, \{S\}, P\}$

set of variables — Terminals — start symbol — Production rules

where P = production rules are defined below –

$S \rightarrow SS$

$S \rightarrow 1S00$

$S \rightarrow 00S1$

$S \rightarrow 0S1S0$

$S \rightarrow \varepsilon$

OR

$S \rightarrow SS \mid 1S00 \mid 00S1 \mid 0S1S0 \mid \varepsilon$

Every rule of this CFG produces string with twice as many 0s as 1s. Thus this is our required CFG.

Proof

- The empty string is valid in our language and can be derived by $S \rightarrow \varepsilon$.
- Let us now define a function score (string) which is equal to no. of zeroes – (2×no. of ones) i.e

score (string) = no. of zeroes – 2× no. of ones (in that string)

Thus for all strings in our language score(string) = 0

Let us prove by induction

- Let's assume that for all strings $|s| < n$ can be produced for some $n \geq 0$
- Let $|s| = n$ be any string present in our language.

(1.) If s can be written as a combination of 2 strings $s = ab$ such that score(a) = 0, then score(b) = 0 because score(a) + score(b) = 0 as s belongs in our language. Thus, such strings can be derived using

$S \rightarrow SS$

(2.) Lets consider all possible proper nontrivial prefixes, p of a string s such that score(p) > 0 then they must begin with 00. Since score(s) = 0, and score of $(s_1, s_2 \ldots s_{n-1})$ is negative if $s_n = 0$ where (n = length of s)

thus s_n must be 1. ∴ this string could be written as $00s'1$ and can be generated using

$S \rightarrow 00S1$

(3) Similarly if we consider proper nontrivial suffixes p of s such that score(p) < 0. then these types can be made by $S \rightarrow 1S00$

(4) Let us consider some i such that score$(s_1 s_2 \ldots s_i) > 0$ and score$(s_1 s_2 \ldots s_{i+1}) < 0$ and no nontrivial

prefix a exists such that score(a) = 0, then 3 inferences can be made

(i) $s_{i+1} = 1$

(ii) score $(s_1 s_2 \ldots s_i) = 1$

(iii) string s start with 0.

similarly for string $(s_{i+2} s_{i+3} \ldots s_n)$

(i) score $(s_{i+2} s_{i+3} \ldots s_n)$ = score (w) − score (s_{i+1}) − score $(s_1 s_2 \ldots s_i)$

$= 0 - (-2) - 1 = 1$

(ii) string $s_{i+2} s_{i+3} \ldots s_n$ must end in 0.

Hence $s_2 s_3 \ldots s_i = s_{i+2} s_{i+3} \ldots s_{n-1} = 0$

Such strings can be easily derived using

| S → 0 S 1 S 0 |

Thus our CFG covers all types of strings in our language and hence is valid.

PDA $\Rightarrow M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$

$M = (\{q_0, q_1, q_2 q_3 q_4 q_5\}, \{0,1\}, \{0,1,Z\}, \delta, q_0, Z, q_0)$

$\delta \Rightarrow$

$\delta(q_0, \lambda, \lambda) = (q_1, Z)$

$\delta(q_1, \lambda, Z) = (q_0, \lambda)$

$\delta(q_1, 0, \lambda) = (q_2, 0)$

$\delta(q_1, 1, \lambda) = (q_5, 0)$

$\delta(q_2, \lambda, Z) = (q_1, Z)$

$\delta(q_2, 0, \lambda) = (q_2, 0)$

$\delta(q_2, 1, 0) = (q_4, \lambda)$

$\delta(q_3, \lambda, Z) = (q_1, Z)$

$\delta(q_3, 0, 0) = (q_3, \lambda)$

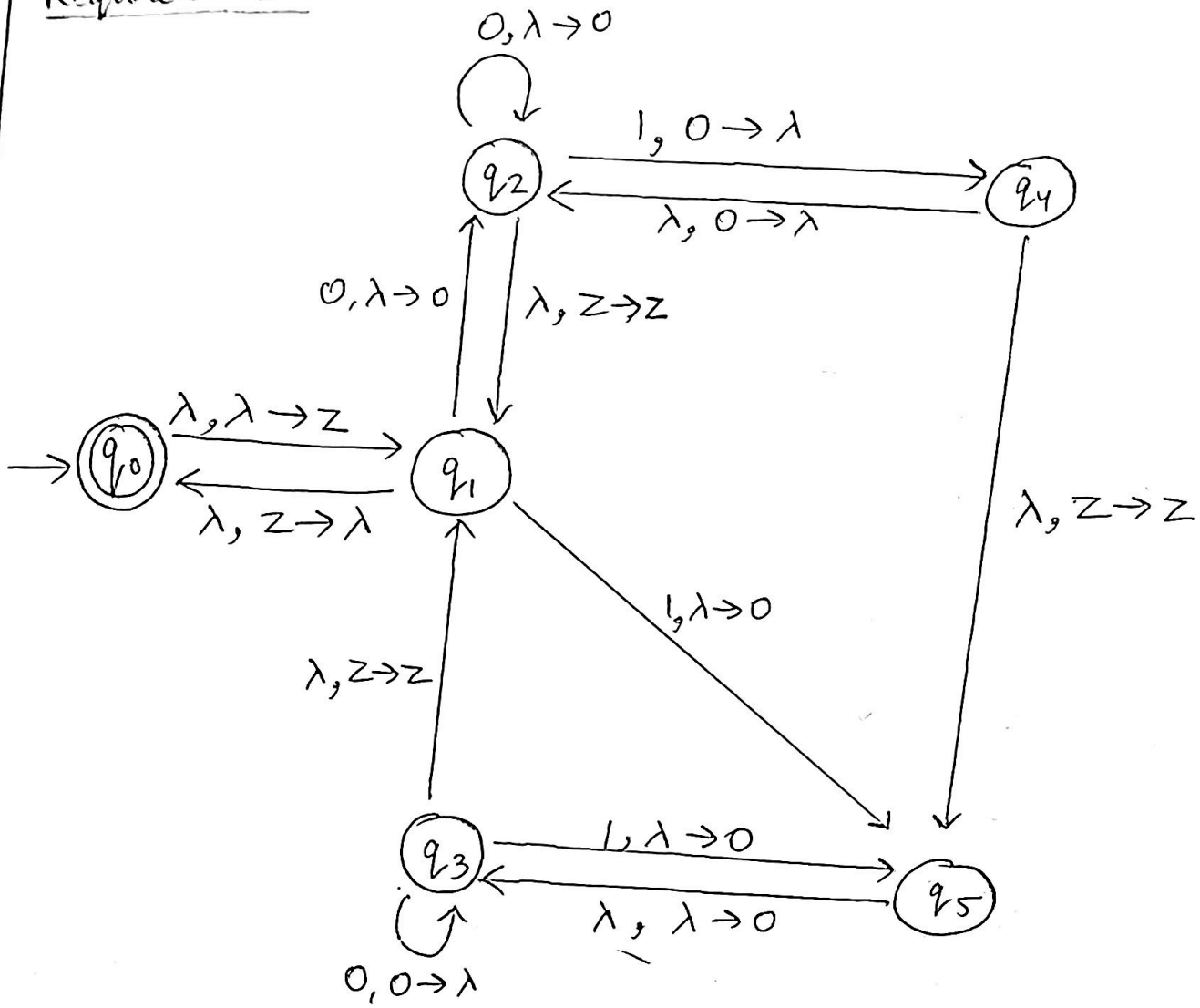
$\delta(q_3, 1, \lambda) = (q_5, 0)$

$\delta(q_4, \lambda, 0) = (q_2, \lambda)$

$\delta(q_4, \lambda, Z) = (q_5, Z)$

$\delta(q_5, \lambda, \lambda) = (q_3, 0)$

Required PDA



This PDA accepts a string if it reaches q_0 with an empty stack or with the start symbol Z at q_1.

(b) To prove that following language is content free:

$L = \{ s_1 s_2 \ldots s_n t_1 t_2 \ldots t_n \mid s_i \in L_1, t_i \in L_2, n \in N \}$ where L_1 and L_2.

Let the grammar of L_1 be $\{ V_1, T, S_1, P_1 \}$ and that of L_2 be $\{ V_2, T, S_2, P_2 \}$. We can rename the variables of differently so that no two variables name in L_1 and L_2 are same.

Let us consider a CFG given by $\{V, T, S, P\}$ where $V = V_1 \cup V_2 \cup \{S\}$ where S is new start state

$$T = T_1 \cup T_2$$

$$S$$

$$P = P_1 \cup P_2 \cup \{ S \rightarrow S_1 S S_2 \mid \varepsilon \}$$

Thus only one new rule is introduced that is

$$S \rightarrow S_1 S S_2 \mid \varepsilon$$

S_1 derives a string in L_1 — S_2 derives a string in L_2

→ also the order S_1 then S_2 is maintained in any subsequent productions due to its structure.

→ Since this is the only source of production in $S_1 S S_2$, then the no. of strings from each languages are also equal to some $n \in N$.

→ This rule will either give us an empty string ε or $S_1 S S_2$. Thus the strings produced by this CFG are exactly those contained in our given language.

→ It follows all the rule of CFG as $(S \rightarrow S_1 S S_2 \mid \varepsilon)$ is correct and rules for S_1 and S_2 are already Content-free. Hence this is also a CFG.