Name - Varun Gupta

Roll No - 2023101108

Course - Automata Theory

Theory Assignment - 1

A3) Making Transition Table;

	0	1
→ 0	1	8
1	2	4
2	8	3
3	2	8
4	5	6
5	8	6
(6)	7	8
7	7	7
8	8	8

Using Partition Method to Minimize

0-Equi:- {0,1,2,3,4,5,7,8} {6}

1-Equi:- {0,1,2,3,7,8} {6}
{4,5}

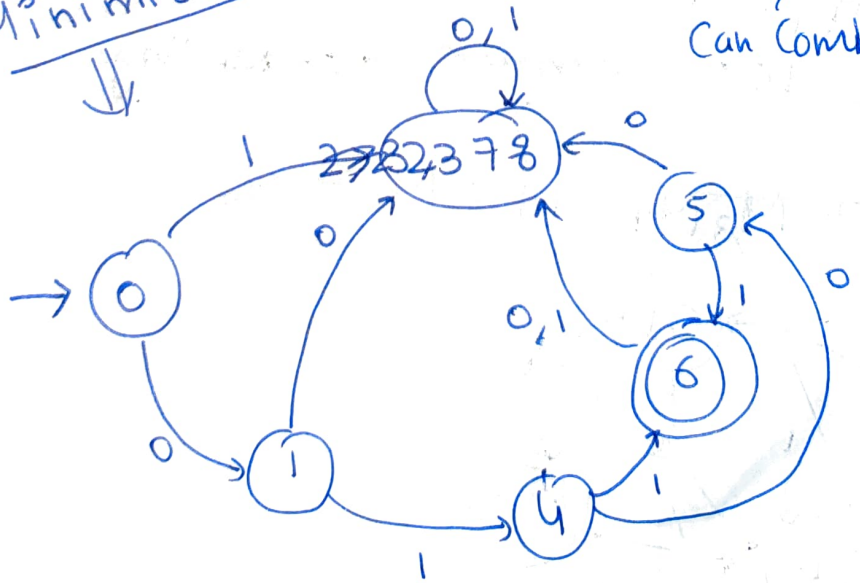
2-Equi:- {4} {5} {6} {0,2,3,7,8}
{1}

3-Equi:- {0} {1} {4} {5} {6}
{2,3,7,8}

4-Equi:- {2,3,7,8} {0} {1} {4}
{5} {6}

↑ Can Combine

Minimized DFA



A1) Alphabet has 3 symbols. & Taking 'ε' into account we have 4 options (3 symbols & ε)

Each state can transition to any of the 5 state with 4 options giving total transition as $5 \times 4 = 20$.

Now we have a total of 5 states and each state can have these much transitions

$\therefore 5 \times 20 = 100$ transitions.

$\Rightarrow$ The maximum number of transitions the NFA can have is 100. ∎

A2) We have to construct a transducer that takes as input a program & writes in the output of part that is not commented.

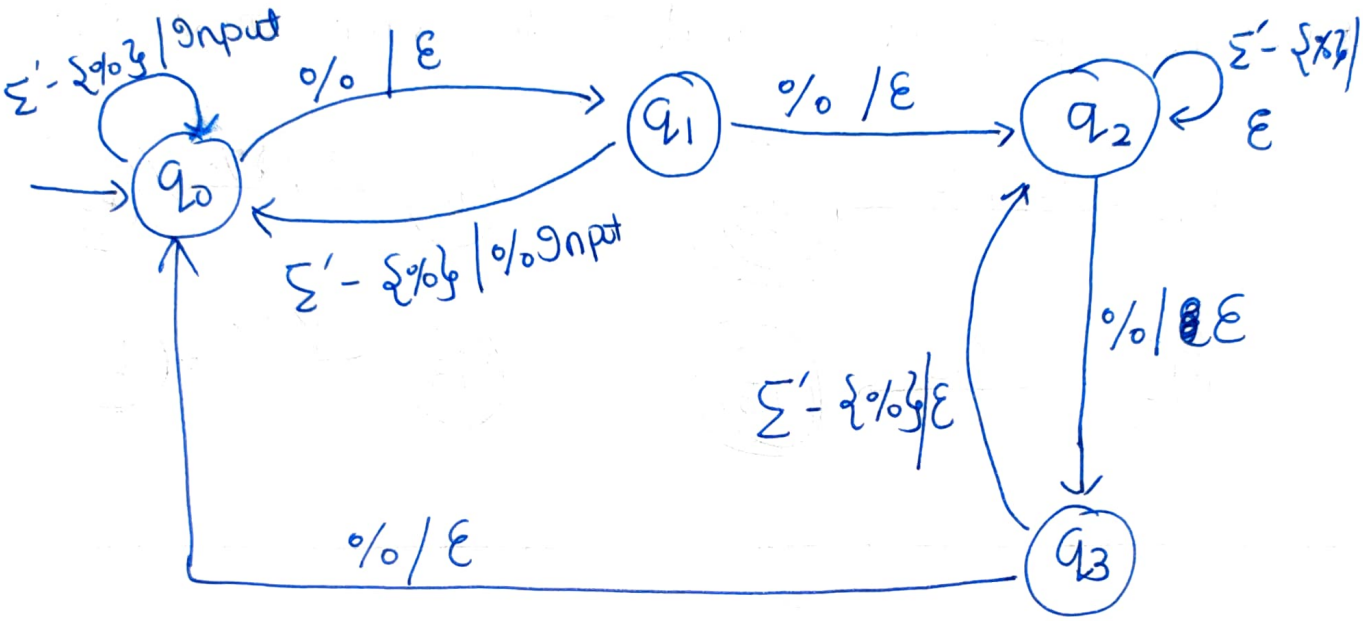
$\Sigma^* \longrightarrow$ set of unicode characters

$\Sigma' \longrightarrow \Sigma \cup \{\varepsilon\}$

Similarly, $\Gamma \longrightarrow \Gamma \cup \{\varepsilon\}$

Let's now define ε. For input 'ε' is treated as Null string. Assuming we will not get any error Character which is not inside of unicode characters.

for output, 'ε' is defined as empty string. that is nothing printed on terminal. 'Input' is defined as input scanned. '%o Input' is concatenation of %o with Input. We define transition as Input / Output. Also defining difference operation i.e $\Sigma' - \{\%\}$ is like everything in Σ' except $\{\%\}$.



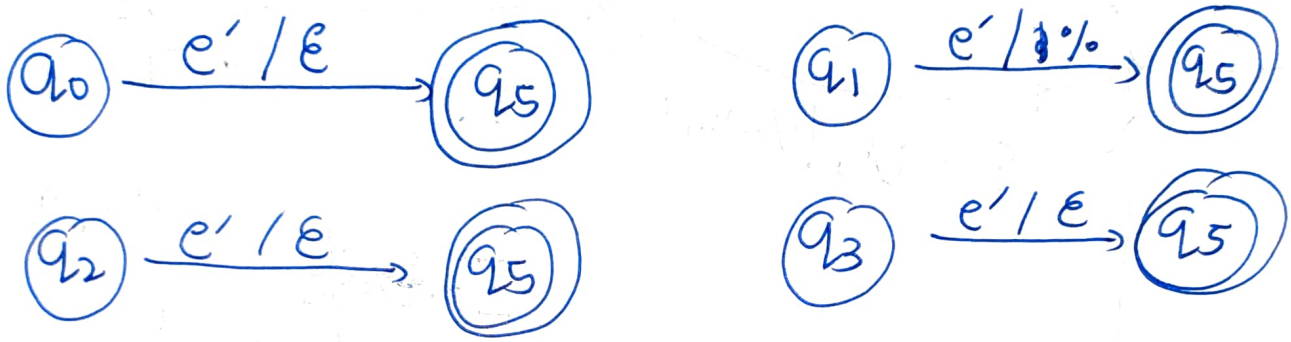
'q_0' is the start state. If we read any input other than $\{\%\}$ we will directly print it. If we get '%' we trasit to q_1 without printing anything. If we get any other input we will print that input with '%' concatenated at start and go back to q_0. If again got '%' we will print nothing and go back to q_2. This is where comment is started now nothing will be printed until "%%" is present. from q_3 if again got "%"

it will end comment and go to q_0 state.

Here there is a ~~caveat~~ Caveat of End of file. So we have not defined anything which depict EOF. Suppose let's say we can scan EOF and is depicted as e'.

I am writing EOF transition from every state.

Also Let 'q_5' is the final state which terminate



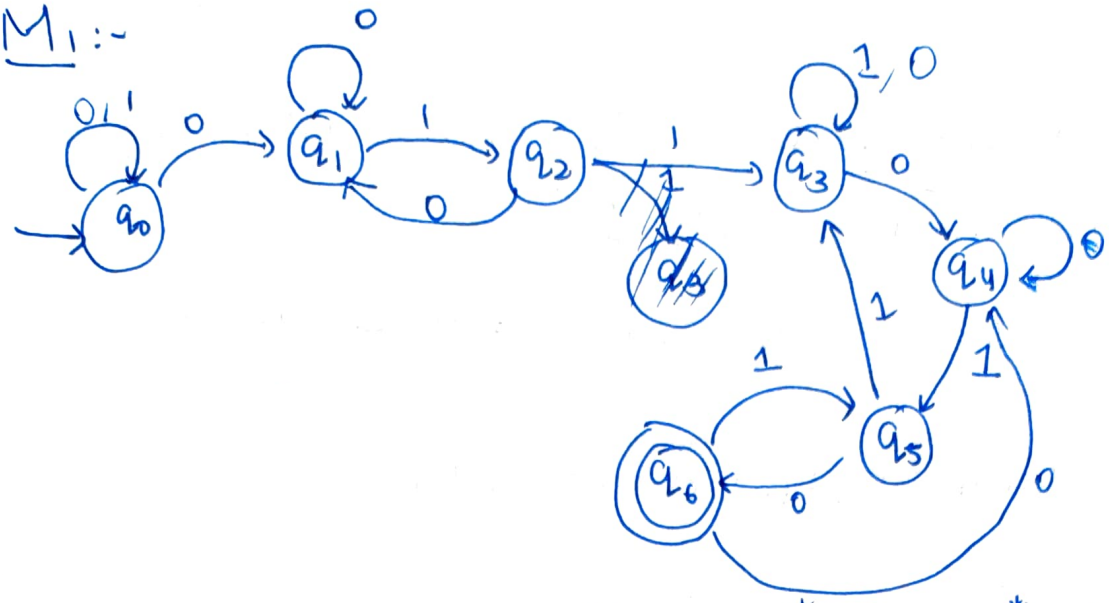
A4) So Let's consider 2 NFAs such that strings

M_1 = end in 010
and
have 011 somewhere preceding

M_2 = end in 101
and
have 100 somewhere preceding

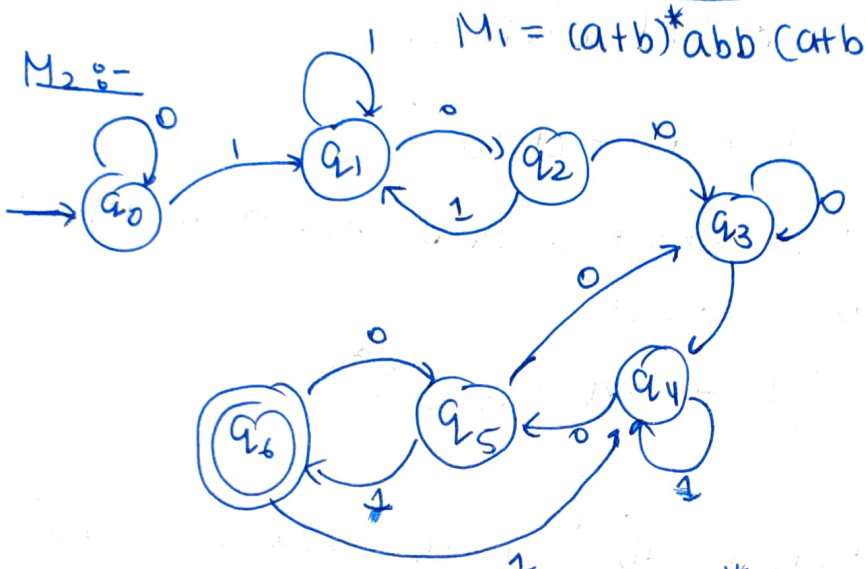
$M = M_1 + M_2$

M_1 :-



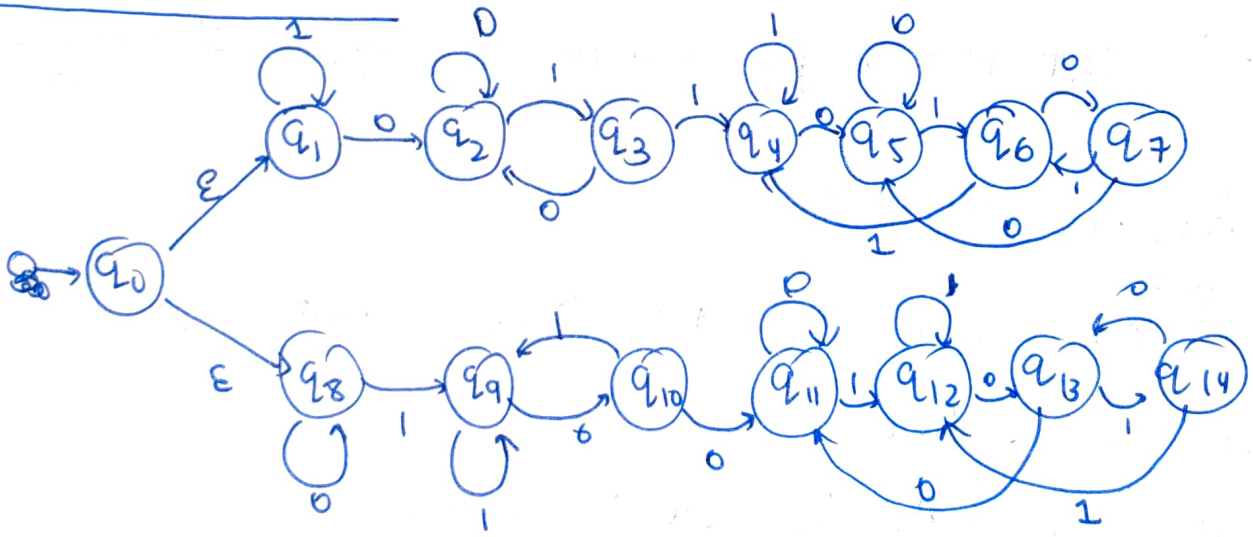
$M_1 = (a+b)^* abb\ (a+b)^*\ aba$

M_2 :-

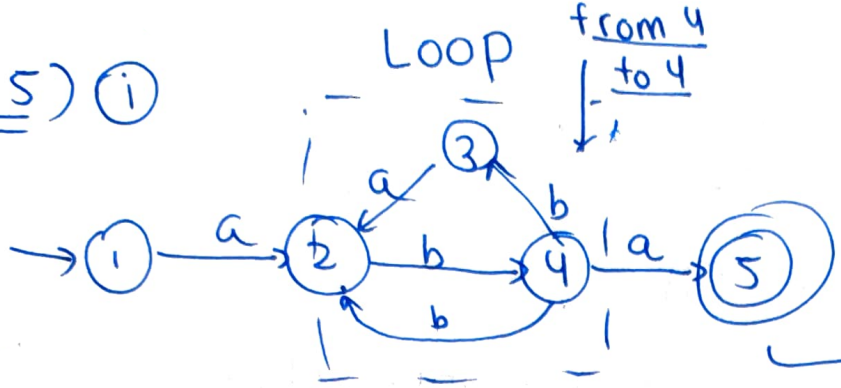


$M_2 = (a+b)^*\ baa\ (a+b)^*\ bab$

Now $M = M_1 \cup M_2$



A5) (i)

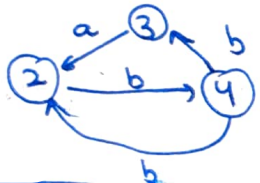


$ab(Loop)^*a$

(upto 4)

$$ab(bb + bab)^* \cdot a$$

This is the regular expression of the language accepted by the provided finite state automata.



There are two loops here (2 to 4), one start at 2 4 i.e $(bb)^*$ & other at $2(bab)^*$. It can loop any no of times

A5)

(ii) As cleared in doubts by TAs that overlapping of substring is allowed that is

-aaaa-- is considered as 2 substrign 'aaa'

RE = (Prefix) aaa (Postfix)

Prextix must end with b.

Prefix = $(b + ab + aab)^*$

Postfix must start with b and d should not contain another 'aaa'

Postfix = $(b + ba + baa)^*$

RE = $(b+ab+aab)^*$ aaa $(b+ba+baa)^*$

A6) Let A = xyz where $|x| = |y| = |z| = n$ (Let assume)

$\therefore A^{1/3} - {}^{1/3} = xz$

Proove:

A is regular $\not\Rightarrow$ $A^{1/3} - {}^{1/3}$ is ~~not~~ regular

Let A is regular

Suppose $A^{1/3} - {}^{1/3}$ is also regular.

Let pumping length, $p = n$.

$A^{1/3} - {}^{1/3} = xz$

By pumping lemma, $A_{1/3} - {}_{1/3} = abc$

Also, $|ab| \leq p$ & $ab^i c \in A_{1/3} - {}_{1/3}$

Since $|x| = |y| = p$

~~a~~ b will contain elements of x.

~~Suppose~~ $\therefore |a| < p$ & $|b| \leq p$

also $|a| + |b| \leq p$

$|c| \geq p$

$(abc)^*$ $|x'| = n$

$n \geq 0$

Let A be $\{$ $\}$ $= x'y'z'$

By pumping lemma, we know this is Regular

Because the total length is always 3k.

Now $A = \{\varepsilon, abc, abcabc, ----\}$

$\therefore A_{1/3} - A_{1/3} = \{\varepsilon, ac, abbc, abcabc, ---\}$

it is of form $x'z'$.

~~By Pumpin~~ Let assume $A_{1/3} - {}_{1/3}$ to be regular.

$\therefore$ By Pumping lemma $A_{1/3} - {}_{1/3} = xyz$

Ut pumping length ~~be~~ $p = n$.

$|xy| \leq p$ & $|y| > 0$ @

$0 < |y| \leq p$ i.e $0 < |y| \leq p$ also $|z| \geq p$

Since $|y|$ can't be zero $|y| \geq 1$

i.e $|x| < p$ or $|x| \leq p - 1$

By pumping lemma, $xy^i z \in A_{1/3 - 1/3} \forall i \geq 0$

for $i = 0$,

$A_{1/3 - 1/3} = xz$

But ~~let is atm~~ $|x| < p$ & $|z| \geq p$

$\therefore |x| \neq |z|$

$\Rightarrow xy^0 z \notin A_{1/3 - 1/3}$

$\Rightarrow A_{1/3 - 1/3}$ is not regular.

$\therefore$ If A is regular, then $A_{1/3 - 1/3}$ is not neceessarily regular.

A7) Given :- M_1 and M_2 be DFAs with K_1 & K_2 states

$U = L(M_1) \cup L(M_2)$. 'S' is some string s.t $S \in U$.

To Proove: $|S| < \max(K_1, K_2)$

Proof :- For a DFA with K states.
(By Pigeonhole Principle [as discussed in class])
For K states there would be K-1 transitions. following this we get a string with length = K-1. surely there would ~~st~~more Strings with length both less than & greater that K-1. But we can surely say there exist some string U such that

$|U| < K$

$U = L(M_1) \cup L(M_2)$

$\Rightarrow$ Any Language in U will be ~~passed~~ atleast passed by either of M_1 or M_2.

If passed by M_1,

$|s| < K_1$ (By above Result)

Or if by M_2,

$|s| < K_2$

As both are possible we can surely say.

$$|s| < \max(K_1, K_2)$$

Hence Proved.

A8)

(1) $L = \{ w \mid w \text{ has balanced paranthesis} \}$

Let $A = \{ \{^n \}^n \mid n \geq 0 \}$

Clearly $A \subseteq L$.

L is Regular $\Rightarrow$ A is Regular

Suppose L is regular.

∴ String s = $\{^p\}^p$ also belong to L

with p = pumping length.

As |s| > p we can apply Pumping lemma.

$s = xyz$.

as $|xy| \leq p$

y only contain symbols of type '{'.

⇒ xyyz would not be of type D.

i.e $xy^2z \notin D$

This contradicts Pumping lemma

∴ L is not regular.

(2) $L = \{a^{n!} \mid n \geq 0\}$, Assume L is Regular

let string be s be $a^{p!}$ (p > 0)

Clearly as $p! \geq p \Rightarrow |s| \geq p$

'p' be pumping length.

$s = xyz$.

Let consider two strings s, s' ∈ D

defined as, $s = xyz$

$s' = xy^2z$ (By Pumping Lemma s' ∉ D)

$|s| = p!$

$|y| \leq p$

$\therefore |s'| \leq |s| + p$ (because s' just has an extra y)

$|s'| \leq p! + p$

$p \geq 0$ or $p \geq 1$

$p \cdot p! \geq p!$

$p\,p! + p \geq p! + p$

$(p+1)\,p! \geq p! + p$

$p! + p \leq (p+1)!$

$\Rightarrow |s'| \leq (p+1)!$

Also, $p! \leq |s'| \leq (p+1)!$

$\therefore s' \notin L$

This contradicts our Assumption, $\Rightarrow$ L is not Regular.

Q9) We know that variables are symbols that can be expanded further (left side) and Terminals are symbols that are not further Expandable (right side)

R, T, V, X are clearly variables.

S, U, W, Y, Z do not apper on the left-hand side of any, implying they might be Terminal. (acc to G).

A10)

A10) $A = \{a^i b^j c^k \mid i,j,k > 0 \text{ \& either } i=j \text{ or } j=k\}$

$A_1 = \{a^i b^j c^k \mid i,j,k \geq 0 \text{ \& } i=j\}$

$A_2 = \{a^i b^j c^k \mid i,j,k \geq 0 \text{ \& } j=k\}$

$A = A_1 \cup A_2$

For A_1

$S \longrightarrow A \mid S_1 C \mid \varepsilon$

$A \longrightarrow aAb \mid \varepsilon$

For A_2

$S \longrightarrow B \mid aS \mid \varepsilon$

$B \longrightarrow bBc \mid \varepsilon$

For A

$S \longrightarrow S_1 \mid S_2$

$S_1 \longrightarrow A \mid S_1 C \mid \varepsilon$

$A \longrightarrow aAb \mid \varepsilon$

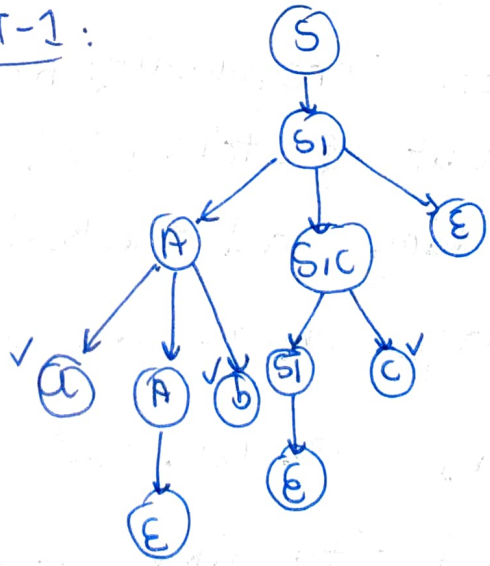
$S_2 \longrightarrow B \mid aS_2 \mid \varepsilon$

$B \longrightarrow bBc \mid \varepsilon$

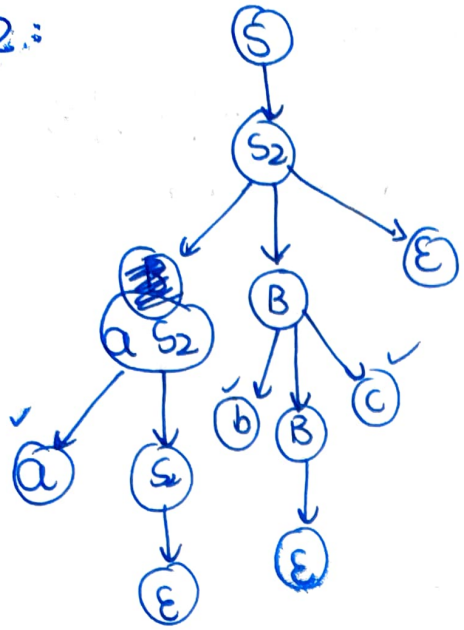
For Ambguity :- The grammar is ambiguous because for $i=j$ & $j=k$ i.e $i=j=k$ of the form $a^n b^n c^n$ we could use ang of S_1 or S_2 to generate the string with different Parse Tree.

let string be abc.

P.T-1 :



PT-2:

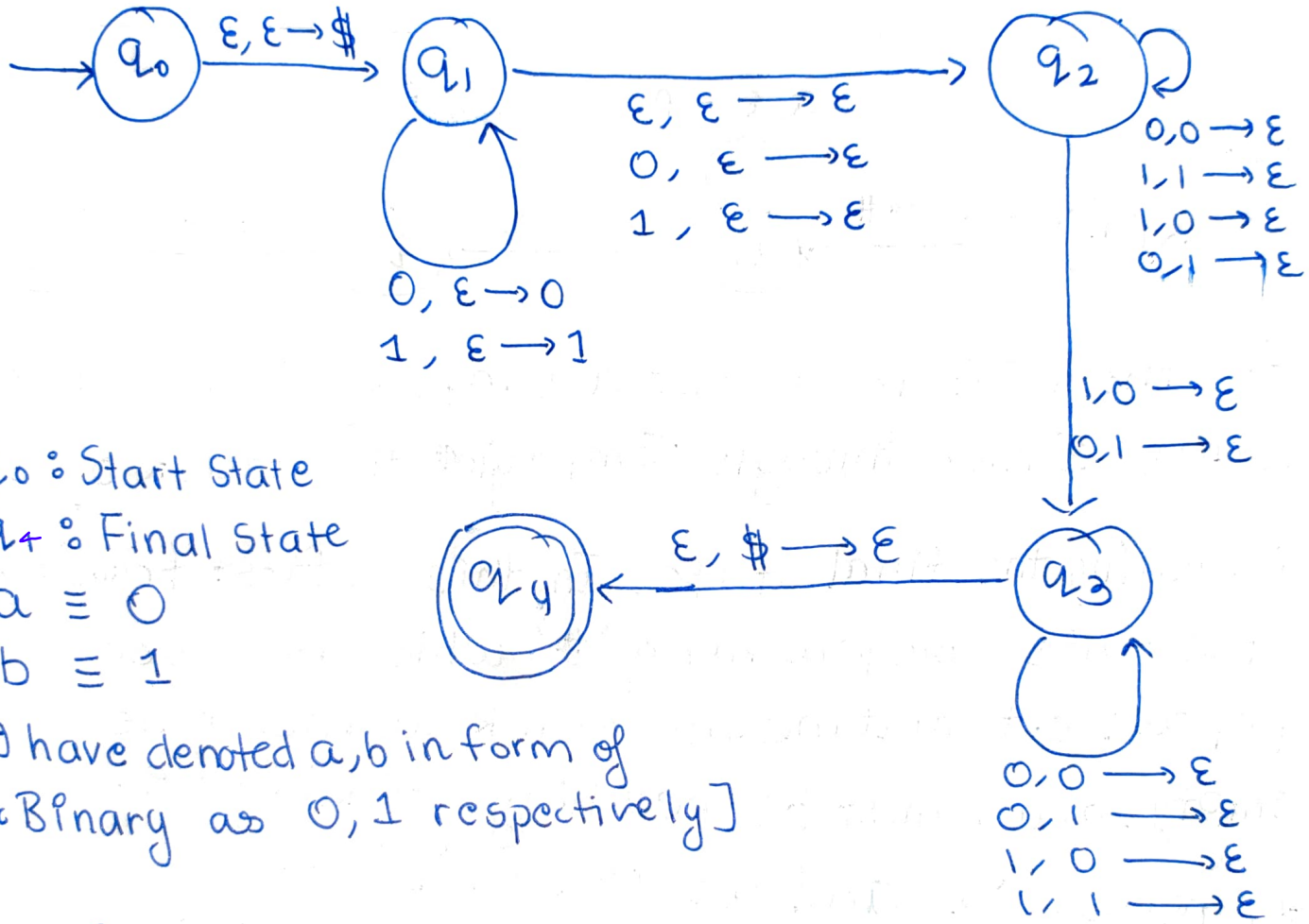


∴ We generated two different Parse Trees for the same String.

A11) We need to construct a PDA for the language of all non-palindromes over {a, b}

As we already have an idea of PDA for palindromes we can just change that to accepts all non-palindromes we just to ~~enforce~~ enforce that ~~that~~ we have atleast one irregularity between first half & second half of string 's'. That is we can keep initial pushing and tackling middle part same and change where we are popping symbols by making sure we get atleast one occurence of irregularity like aab or ba.

I am using Sipser's representation for showing Transistions i.e $a, b \to c$ signify that when machine reading a, it may replace symbol b on top of stack with a c. a, b, c may be ε [Michael Sipser Pg 144] also I \$ is marking used to mark bottom.



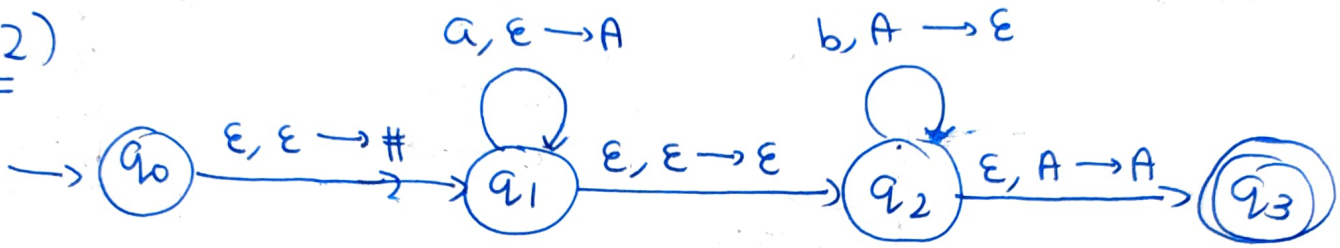
q_0 : Start State
q_4 : Final State
$a \equiv 0$
$b \equiv 1$

[I have denoted a, b in form of & Binary as 0, 1 respectively]

So, $q_0 \to q_1$ is marking bottom with \$. q_1 is pushing all 0, 1 that come into stack. For middle part we use nondeterminism and just take input without doing anything. Then from $q_2 \to q_3$ I am transitioning if I get a mismatch.

Until, any number of matchs & unmatchs are allowed. In q_3 iam sure there is some mismatch already occured and string is not Palindrome. Now in q_3 i am just emptying the stack until bottom mark '$' is reached which transition to q_4 (The final state).

A12)



For the PDA to accept a language it has two ways either through Emptying the stack or by Reaching the final state. In this in first transition itself we are pushing '#' which is never popped out that means stack will never be empty. So we must be reaching final state.

q_1 reads 'a' and q_2 reads 'b'. So Language should of form $\{a^i b^j \mid i,j \geq 0\}$

Now ~~go~~ to reach q_3 ~~from~~ (final state) from q_2. we are popping 'A' & then pushing 'A'. So if stack does not have 'A' in it we

will never reach q_3. Now q_1 is pushing 'A' and q_2 is popping it out. Therefore q_1 should push more 'A' than q_2 is popping to reach final state.

$\Rightarrow i > j$

$L = \{ a^i b^j \mid i > j \;\&\&\; i, j \geq 0 \}$

Let string $s \in L$. (s will by accepted by PDA), [obvious]

$|s| = i + j$

# of string's with length 100 :-

$|s| = 100$

$i + j = 100$

$i = 100 - j$

But, $i > j \Rightarrow 100 - j > j$

$2j < 100$

$\therefore 0 \leq j \leq 49$

For a particular j we have a unique i.

$\therefore$ # of Possible Strings = 50

Q