7 (c) $\bar{L} = \{w \mid w \notin L$ & w does not contain an 'a'$\}$

Let L be regular language & $\bar{L}$ is also regular as complement of L is closed under for complementation

the language can be represented as

$A = (b \cup c)^*$

let A be language over $\{a, b, c\}$ that don't contain a

$\Rightarrow \bar{L} \cap A$

7 (d) $\{uv \mid u \in L$ and $v \notin L\}$

u & v are strings in L & it's complement

$L = \{a^i b^i c^i \mid i \geq 0\}$ $\therefore$ L is regular

$\bar{L} = \Sigma^* - L$ is also regular because since Regular languages are closed under complementation

~~concaten~~ uv represent concatenation of languages $L \cdot (\Sigma^* - L) \Rightarrow L \cdot \bar{L}$

$L \cdot (\{a, b, c\}^* - L)$

(or) we can construct NFA-λ by concatenating L & its complement.

14 (c) using pumping lemma to show that each of the following sets is not regular (2)

$L = \{a^i b^j c^{2j} \mid i \geq 0, j \geq 0\}$

Assume L is regular, pumping lemma holds for ~~[illegible]~~ some k (or) more

$z \in L$, $\forall z = uvw$, $|z| \geq k$

length $(uvw) \leq k$

length $(v) > 0$ (or) $v \neq \lambda$

$uv^iw \in L$ ~~for~~ $\forall i \geq 0$

$z = a b^k c^{2k}$

case 1 : $a \notin v$

$$\underset{ab^i}{u} \quad \underset{b^j}{v} \quad \underset{b^{k-j-i}c^{2k}}{w}$$

$i+j \leq k-1$ & $j > 0$

pumping v recursively uv^2w

$= ab^i \, b^j b^j \, \underset{w}{b^{k-j-i}c^{2k}}$

$a b^i \cancel{b} \, b^k c^{2k}$ which is not in L

because no of b's are more than half of c's

case 2 :

$a \in v$

$$\underset{\lambda}{u} \quad \underset{ab^i}{v} \quad \underset{b^{k-i}c^{2k}}{w}$$

$uv^2w = ab^iab^i \, b^{k-i} c^{2k}$
$= ab^iab^k c^{2k} \notin L$

(or) $i=0$
uv^0w
$= b^{k-i}c^{2k}$
$\in L$

$\therefore L$ is not regular since $a^k b^k c^{2k}$ has no decomposition with pumping ~~lemma~~ string

(d) $L = \{ww \mid w \in \{a,b\}^*\}$

Assume L is regular, pumping lemma holds for some k (or) more

$z \in L$, $|z| \geq k$ & $z = uvw$

let $z = \underbrace{a^k b^k}_{w} \underbrace{a^k b^k}_{w}$

case 1: ~~a~~ ~~a~~ $a \in V$

$\overset{u}{a^i} \; \overset{v}{a^{k-i} b^j} \; \overset{w}{b^{k-j} a^k b^k}$

Consider string uv^2w

$uv^2w = a^i \; a^{k-i} b^j a^{k-i} b^j \; b^{k-j} a^k b^k$

$\Rightarrow a^k b^j a^{k-i} b^j b^{k-j} a^k b^k$

$\Rightarrow a^k b^j a^{k-i} b^k a^k b^k$

$\Rightarrow uv^2w \notin L$

because a's & b's is first w is not same second w ~~is~~

$\therefore L$ is not regular since $a^k b^k a^k b^k$ has no decomposition with pumping string.

2

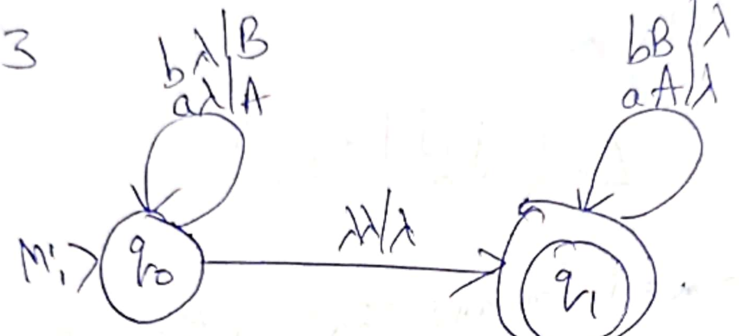
(a) Give the transition table of M

(b) Trace all computation of the strings ab, abb, abbb in M

(c) Show that aaaa, baab $\in L(M)$

(d) Show that aaa, ab $\notin L(M)$

Example 7.1.3



$L(M) = \{ ww^R \mid w \in \{a,b\}^* \}$

(a)

State	~~Transition~~ Nextstate	~~Stack top~~ i/p	stack top
q_0	$[q_0, A]$	a	λ
q_0	$[q_0, B]$	b	λ
q_0	$[q_1, \lambda]$	λ	λ
q_1	$[q_1, \lambda]$	a	Pop A
q_1	$[q_1, \lambda]$	b	Pop B
~~q_1~~			

(b)

(i) $\vdash [q_0, ab, \lambda]$

$\vdash [q_0, b, A]$

$\vdash [q_0, \lambda, BA]$

(ii) $\vdash [q_0, abb, \lambda]$

$\vdash [q_0, bb, A]$

$\vdash [q_0, b, BA]$

$\vdash [q_1, b, BA]$

$\vdash [q_1, \lambda, A]$

(iii)

$\vdash [q_0, abbb, \lambda]$

$\vdash [q_0, bbb, A]$

$\vdash [q_0, bb, BA]$

$\vdash [q_0, b, BBA]$

$\vdash [q_1, b, BBA]$

$\vdash [q_1, \lambda, BA]$

(c) (i) aaaa

$\vdash [q_0, aaaa, \lambda]$

$\vdash [q_0, aaa, A]$

$\vdash [q_0, aa, AA]$

$\vdash [q_1, aa, AA]$

$\vdash [q_1, a, A]$

$\vdash [q_1, \lambda, \lambda]$

(ii)

$\vdash [q_0, baab, \lambda]$

$\vdash [q_0, aab, B]$

$\vdash [q_0, ab, AB]$

$\vdash [q_1, ab, AB]$

$\vdash [q_1, b, B]$

$\vdash [q_1, \lambda, \lambda]$

∴ for c(i) & (ii) the computation ends with
i/p : λ & stacktop is empty
So they $\in L(M)$

(d) aaa, ab $\notin L(M)$

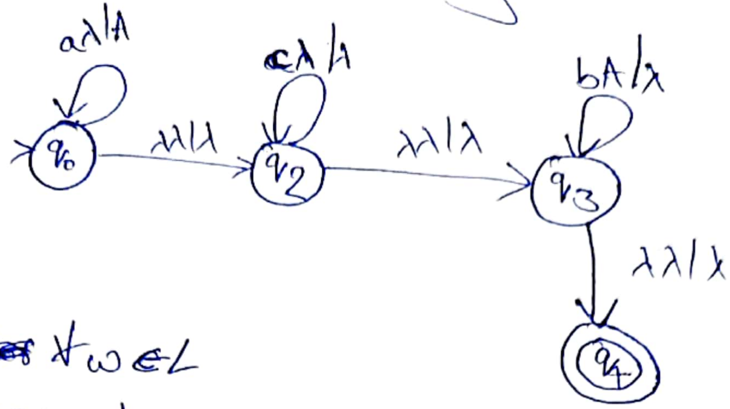
$\vdash [q_0, aaa, \lambda]$
$\vdash [q_0, aa, A]$
$\vdash [q_0, a, AA]$
$\vdash [q_0, \lambda, AAA]$
$\notin L(M)$

$\vdash [q_0, ab, \lambda]$
$\vdash [q_0, b, A]$
$\vdash [q_0, \lambda, BA]$
$\notin L(M)$
because stack is not empty

Scanned with OKEN Scanner

③ ⓑ PDA:

$L = \{a^i c^j b^i \mid i, j \geq 0\}$



~~for~~ $\forall w \in L$

* on reading 'a' from i/p string push 'A' on to the stack
* on reading 'c' from i/p string just skip it (read all c's)
* on reading 'b' from i/p string pop 'A's on the top of the stack

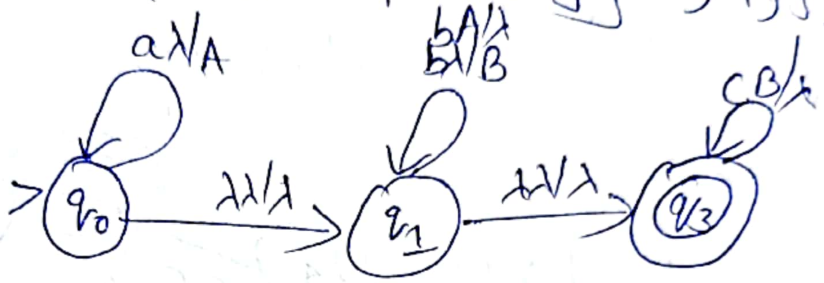
~~pseudocode~~
if w = λ:
  accept it
~~while w ≠ λ~~
else:
  ~~while w ≠ λ:~~
~~for~~

procedure:

```
n = |w|
if n == 0:
    accept it
else:  stack = []
    for i in w:
        if i == 'a':
            // the push 'A' on to stack
```

```
        stack.push('A')
    else if i == c:
            pass
    else :
        ~~stack.pop('A')~~
        ~~Stack~~ stack.pop(); // pop A from stack
end else
```

(c) $L = \{a^i b^j c^k \mid i+k=j\}$, assume $i, j, k \geq 0$



for every 'a' push A on stack

for every 'b' & if stack top has A pop the stack & process

~~for~~ If no ~~A~~ A on stack top push 'B' onto the stack

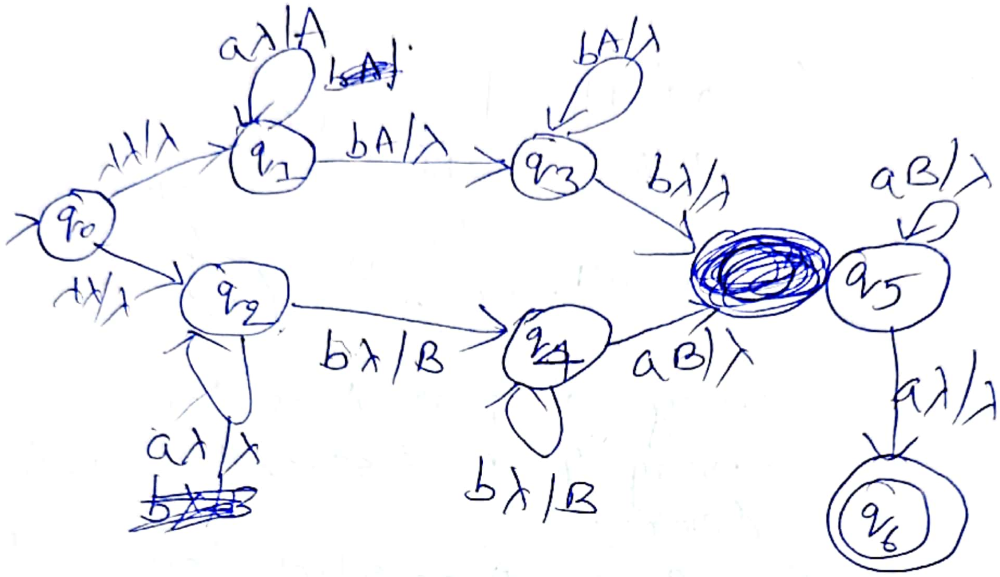
If 'c' is i/p symbol pop B's from the top of the stack

Pseudo Code:

```
n = len(ω)
if n == 0:
    accept
else:
    stack = []
    for i in ω:
        if i == 'a':
            stack.push('A')
        else if i == 'b' & stack.peek() == 'A':
            stack.pop()
        else if i == 'b' & stack.peek() != 'A':
            stack.push('B')
```

else:
    stack.pop()
end if

(9) $L = \{a^i b^j \mid i \neq j\}$



Pseudocode:

input("guess")
if guess is more a's:
    then state = q_1
else:
    state = q_2
n = len(w), Stack = []
for i in w
    if state == q_1 & i == 'a':
        stack.push('A')
    else state == q_1 & i == 'b':

```
if stack is Empty()
        //go to q6 from q1
else:
        state=q3
        stack.pop()
else if   state==q3 & ~~ch~~ i==='b';
        if stack is empty():
            accept
        else
            stack.pop()
else if
        state==q2 & i=='a':
            // more b's than a's
else if  state==q2 & ~~ch~~ i=='b';
            state = q4
            stack.push('B')
else if state==q4 & ~~i~~==b;
            stack.push(B)
else if  state ==q4 & i==a;
            state=q5
        if stack.isEmpty()
            //more a's than b's
            Pass
        else;
            stack.pop()
else if
        state==q5 & i==a;
        if stack.peek()==λ;
            Pass
```

```
else :  pass
```