305 — TOA — 20/8/24

learn about theoratical CS (abstraction)

computer
↓
intelligent machine for which lowest level of intelligence is decision making
↙ acceptance ↘ rejection

collection of unique elements. (set)

C++ has words — belongs to classes i.e.

In compiler, each and every thing is a word that belongs to a specific class

- Keywords (finite set) of reserve words.
- finite Operators (assignments, comparators)
- finite punctuations (brackets, semi-colon)
- variables / identifiers (set of alphabets) → infinite
  - always finite
  - {0-9, a-z, A-Z, _} valid set. (10, 26, 26, 1)
  - cardinality = 63

* finite set, so cardinality can be defined.

* words is nothing but a collection of alphabets.

string n@me = "Umer";
invalid word → rejection. (because not in valid set).

word tokenization (word isolation)

↳ in english, its in basis of spaces.

```
if ( a < s) {
    cout<<" very nice";
}
```

14 words.

int abc=10;
int abc = 10;
(5 words)

⇒ name of variable is not limited (infinite)

set of variables is infinite in compiler

⇒ words belongs to languages (e.g C++) → set of words. which further belong to sublanguages.

e.g keywords, punctuation etc.

Finite → exact word matching
$|\mathcal{L}| = ?$

Infinite → pattern identification.
e.g phone number format 03— ——

C++ language = { keyword, operators, punctuation, identifiers } → infinite (finite)

answer in manner → smaller length words first
• starting with a

language $\mathcal{L}$ = { a, aa, ab, aaa, aab, aba, abb, ... }

Set of alphabet Σ = { a, b }

Question $\mathcal{L} = \{ x \mid x \in \Sigma^{*} ; x \text{ starts with "a"} \}$ (* → length of word)

$\mathcal{L} = \{ a \}$ * = 1
$|\mathcal{L}| = 1$

if * = 2
then $\mathcal{L} = \{ a \}$ is wrong because word length is 2

$\mathcal{L} = \{ aa, ab \}$
$|\mathcal{L}| = 2$

cross-product
$(x \in \Sigma . \Sigma)$ Σ^2
{a, b} . {a, b}
aa bb ab ba
$|\mathcal{L}| = 4$
without condition.

$\Sigma^{*} = \boxed{\Sigma^0} \cup \Sigma^1 \cup \Sigma^2 \ldots$ (length of word = zero)
any length.

$\Sigma \cdot \Sigma \cdot \Sigma$ Σ^3 $|\mathcal{L}| = 8$

$\Sigma^0 = \{ \in \}$ emptystring or string of length 0.

→ encouraged.

Canonical Order: write language in increasing order else unable to detect pattern.

Lexographical order discouraged

Σ^0 can written as λ, π, ϵ

word x = name $\Sigma = \{a, n, me\}$

$|x| = 3$

$L = \{$ name , n , a , nameϵ $\}$

↓ not in set of alphabet.

Regular languages.

↳ creation / membership.

to validate languages, we have ① (machines) (finite automata)

② → (expressions.) (regular expressions)

22/8/24

Language → collection of words

words → collection of alphabets.

$L = \{ x \mid x \in \Sigma^* \}$; $|L| = 10$ $\Sigma = \{0, 1\}$

λ,

$L = \{ 0, 00, 000, 0000, 00000 \}$ → lexographical order which creates problem in identifying pattern.

cannot skip word of length 1, 2 but can skip 3 cause

$L = \{ 0, 1, 00, 01, 11, 10, 000, 001, 010 \ldots \}$ set is of 10 elements

↳ canonical order.

λ (should start with null)

empty intialization is word of length zero

↱ upperbound $\Sigma^0 \cup \Sigma^1 \cup \Sigma^2$

$L = \{x \mid x \in \Sigma^* ; |L| \leq 10 \text{ & } |x| \leq 2\}$

$L = \{\wedge, 0, 1, 00, 01, 10, 11\}$ $\boxed{|L| = 7}$

$L = \{x \mid x \in \Sigma^* ; |L| \leq 10 \text{ & } |x| = 2\}$

Σ^2

$\Sigma \cdot \Sigma$

$L = \{00, 01, 10, 11\}$

$\boxed{|L| = 4}$

⇒ ~~Kleene star~~ ~~Kleene closure~~

* clean star (clean closure)

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots$

(+) $\Sigma^+ = \Sigma^* - \Sigma^0 = \Sigma^1 \cup \Sigma^2 \ldots$

we are taking about length 1 and ownwards

even if the context is same like 0 and 00, but as a word they are different in a language as it depends upon their length

$L = \{x \mid x \in \Sigma^* ; |L| \leq 10 \text{ & } |x| \leq 2 \text{ & } x \text{ starts with } 0\}$

$L = \{0, 00, 01\}$

$\boxed{|L| = 3}$

→ we write language because of pattern recognition.

$L = \{ 0, 00, 10, 000, 010, 100, 110, \ldots \}$

$\Sigma = \{0, 1\}$

$L = \{ x \mid x \in \Sigma+ ;$ x ends with '0' $\}$

; x does not end with '1'

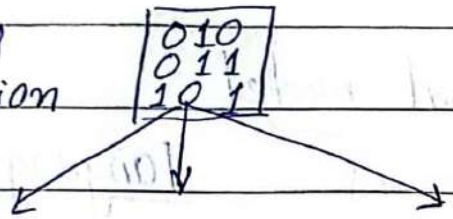
; x as a decimal number divisible by 2

" by 5

" by 10

" mod of 2 is 0

conceptual decomposition



Prefix L→R | Substring in between | suffix R−L

→ if it contains 0, 01 should be there

maybe start, end not mandatory.

$\overrightarrow{100}$ $\overleftarrow{100}$

Prefix		suffix
L=0	∧	∧
L=1	1/00	10/0
L=2	10/0	1/00
L=3	100/	/100

$L = \{ x \mid x \in \Sigma^* ;$ x as a decimal number number divisible by 2 & binary number $\}$

& $|L| = 8$

$L = \{ 0, 00, 10, 000, 010, 100, 110, 0000 \}$

→ ordering inside length is flexible

$L = \{x \mid x \in \Sigma^*$; x as decimal number is divible by 2 & binary numbers but x is as a decimal number not divisble by 4 ; $|L| = 7\}$

$L1$ — $L2$ — $L3$

$L = L1 \cap L2 \cap L3$

$L = \{10, 010, 110, 0010, 0110, 1010, 00010\}$

27/8/24

Machines → mathematical model

class.

Language (regular)

we have (3) types of FA:

(a) Deterministic FA

(b) Non-Deterministic FA (ambigious)

(c) NFA − λ with Null.

(regular) 1. Expression

" 2. Grammer

finite automata. 3. Machine (finite) State machine FSM

FA

↓

(a) DFA

(b) NFA

(c) NFA-λ

Finite Automata (Deterministic)

↓ (5) tuples.

FA

1) Σ [set of alphabets]
2) Q set of states.
3) q_0 start state $q_0 \in Q$
4) f_n final state $f_n \subseteq Q$

↓ [ ] set of final state (can be empty, have 1 or all)

* 5) Transition Function $\delta_{(DFA)}$ $Q \times \Sigma \rightarrow Q$

Notations and Representations:

State (A) → read null

Start state (→S) it has a arrow which does not comes from any other state.

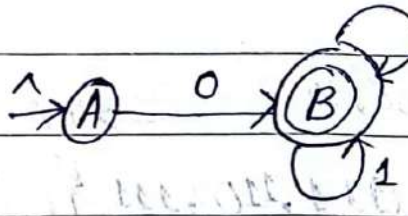
Transition (A) —Σ→ (B) → Σ^1 (power 1) so it read one character.

(A) —0→ (A)

Final state (F)

* finite automata is basically directed graph

$\Sigma = \{0, 1\}$ $Q = \{A, B\}$



* final state can be empty, there might not be any (0).

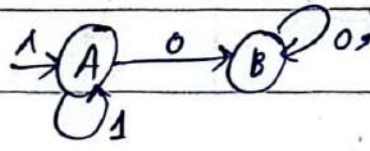
$(A, 0) \to B$ $\quad Q \times \Sigma \to Q$.

$(A, 1) \to ?$ either A / B (so it's not a DFA)

$(B, 0) \to B$

$(B, 1) \to B$

→ machine that accepts null

if 

it is DFA

outdegree
A = 2
B = 2.

at every state, there should be valid transition for every alphabet.

valid DFA: Out degree (→) = no of elements in Σ

1) process
2) accept

Q#: $L = \{x \mid x \in \Sigma^*\}$ $\Sigma = \{0, 1\}$

Construct a DFA for a given language.

First word+1 = length = $\Lambda+1$ = 1 state

$L = \{\Lambda, 0, 1, 00, 01, 10, 11, 000, 001, 010, \ldots\}$

complete DFA for Σ^*

outdegree = 2 = no of Σ

Q#:

$L = \{x \mid x \in \Sigma^*$; x contains '1' as a substring$\}$

$\Sigma = \{0, 1\}$

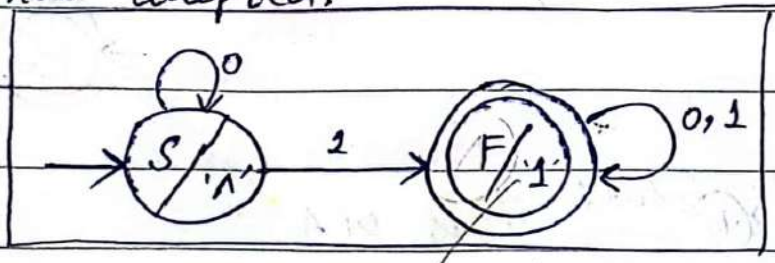
1+1 = 2 states

$L = \{1, 01, 10, 11, 010, 011, 100, 101, 110, 111, \ldots\}$

001

⇒ FA movement is character by character (uni-directional) left → right

(S) we won't make it final so null is not accepted.



100, 101, 110, 111

Σ^2

01, 10, 11

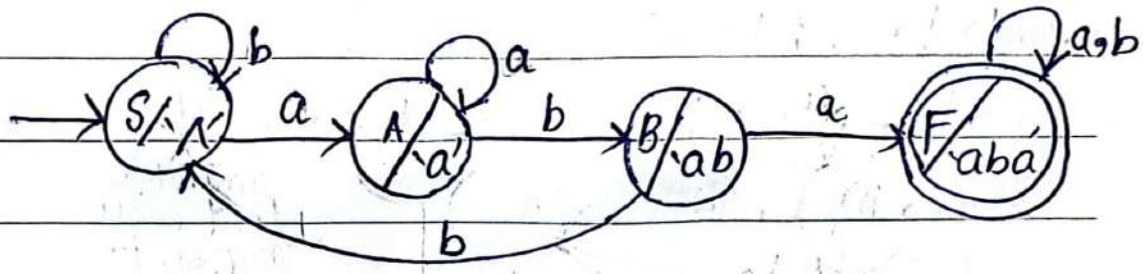
Σ^1

DFA :- outdegree = 2 $|\Sigma| = 2$

↓ name the one that mandatory

Q#: $L = \{x \mid x \in \Sigma^* ;$ x contains 'aba' as a substring $\}$ $\quad \Sigma = \{a, b\}$

$L = \{$ aba, aaba, abaa, baba, abab, aaaba, ababa, baaba, bbaba, ... $\}$



ab → a aba
ab → b abb (pattern destroyed)

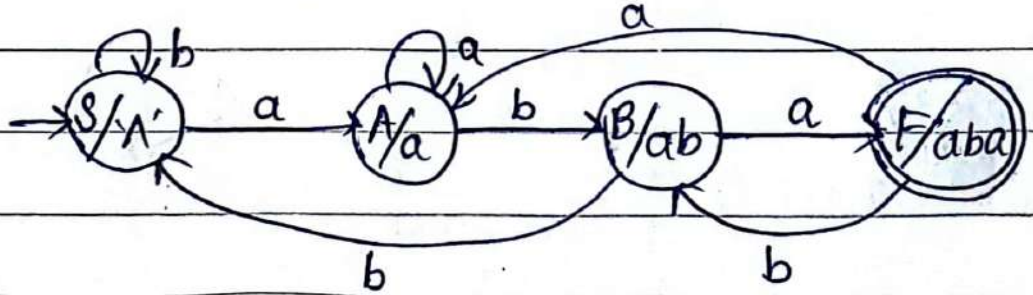
→ whenever we have a pattern, self-loop & accept

$L = \{$ aba, aaba, baba, aaaba, ababa, baaba, bbaba, aaaaaba, aababa, ... $\}$

: x end with 'aba'

↓

if length increase, there may be many problems.



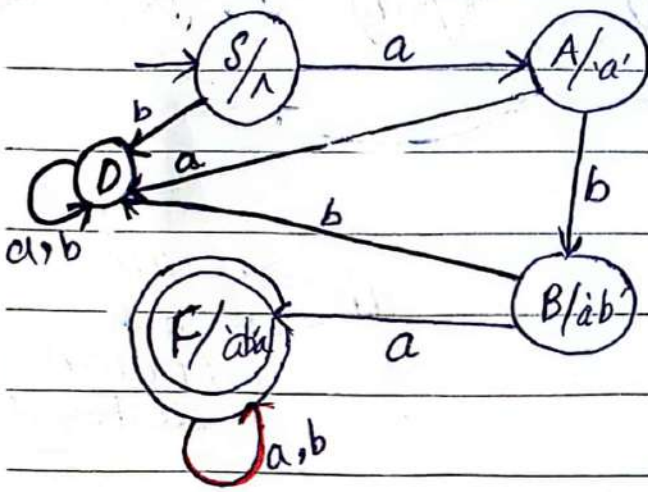
Q#:

29/8/24 $L = \{x \mid x \in \Sigma^* ;$ x starts with 'aba' $\}$

$\Sigma = \{a, b\}$

$L = \{$ aba, abaa, abab, abaaa, abaab, ababa, ababb, ... $\}$

States = 3+1 = 4



Deadstate: any state that will not go to final.

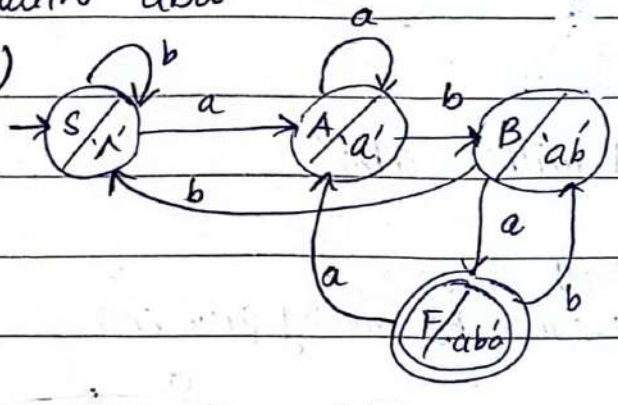
DFA: out-degree = no of alphabets.

method → or contains.

Q# ; x ends with 'aba'

Prefix (pattern)

$L=0$	λ	S
$L=1$	a	A
$L=2$	ab	B
$L=3$	aba	F



Suffix (string)

b match. (so it will go to S)

aa ^ a (longest match on A)

aa

λ — a ✓

b (divide in suffix)

a — a aa

b ab ✓

0 λ

1 b

2 aa

3

ab — a aba
b abb

abb

	suffix (S) match
0	λ
1	b
2	bb
3	abb

aba — a abaa
b abab

discard → pattern length = 3.

abaa

	λ	A
0	λ	
1	a	A
2	aa	
3	baa	

abaa

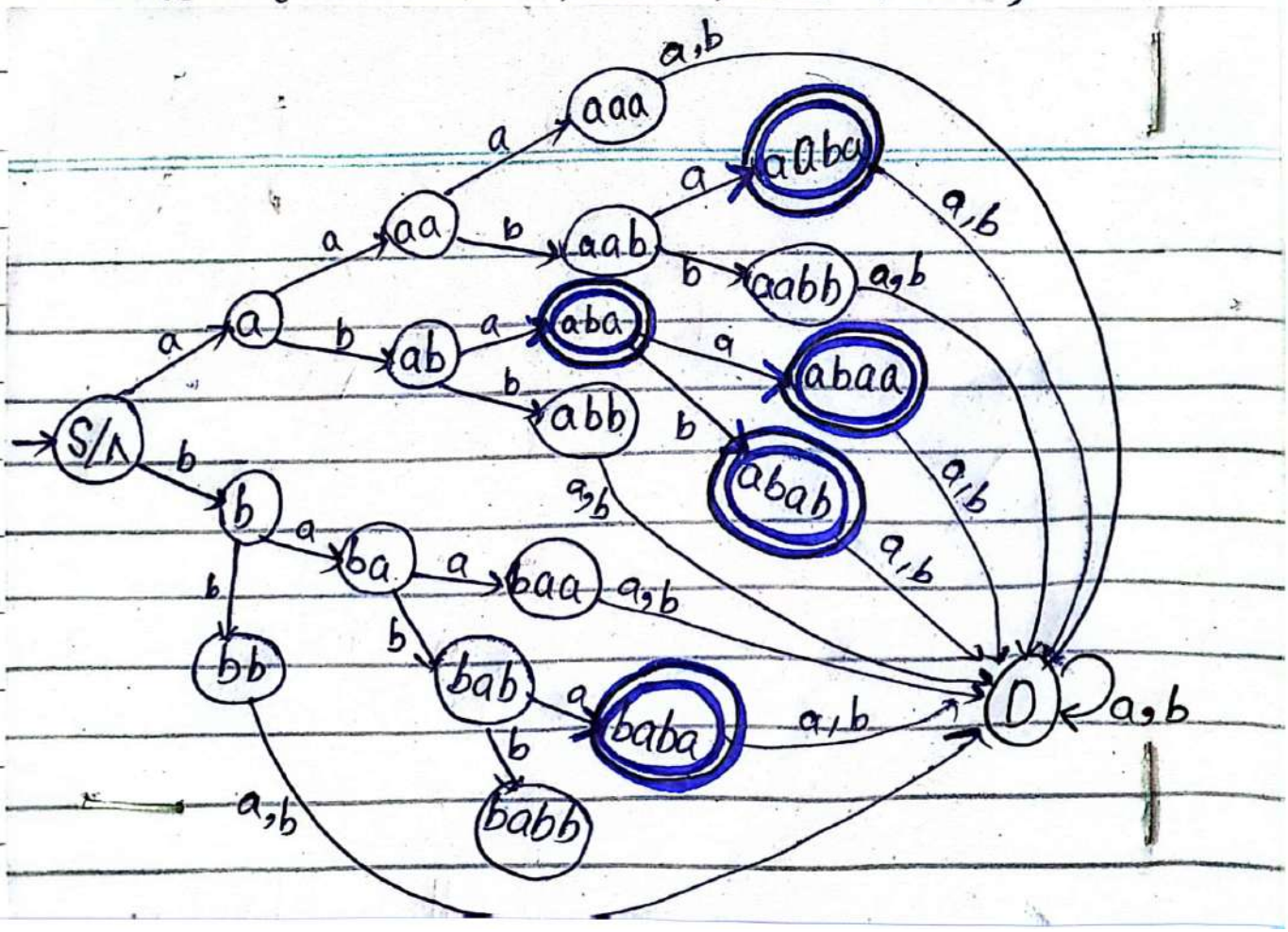
abab

	λ	B
0	λ	
1	b	B
2	ab	
3	bab	

abaa

Q# $L = \{ x \mid x \in \Sigma^*$; x contains 'aba' & $|x| \leq 4 \}$

$L = \{$ aba, aaba, abaa, abab, baba $\}$



* at least $(aa)^1$ lower bound
* exactly $(aa)^1$

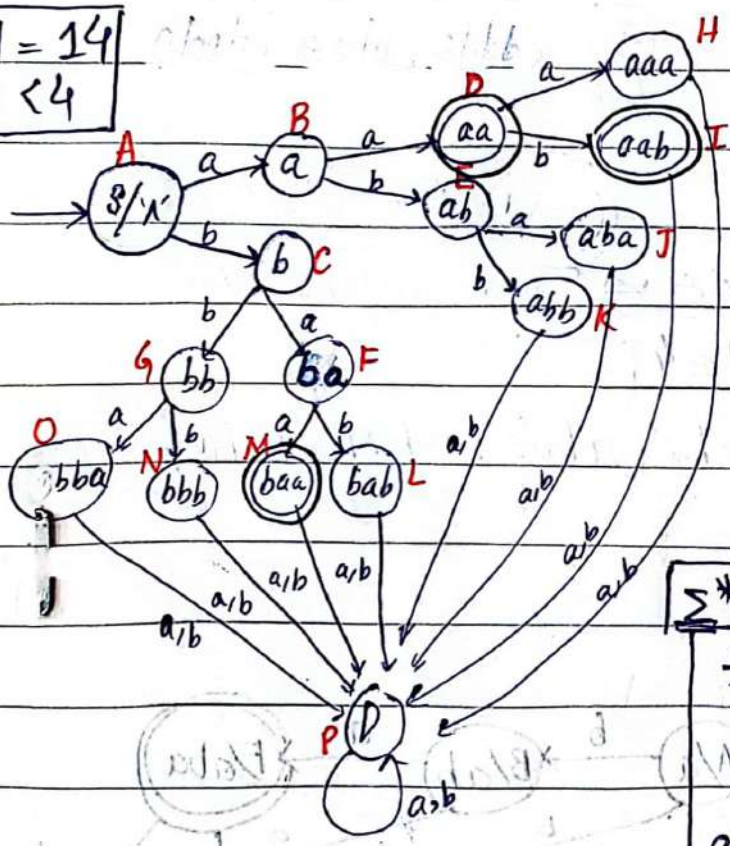
$(aa)^0$ $(aa)^1$ upper bound

Q# $L = \{x \mid x \in \Sigma^*$; x contains at most one occurrence of 'aa' & $|x| < 4\}$

$\Sigma = \{a, b\}$

$L = \{\lambda, a, b, aa, ab, ba, bb,$
$aab, aba, abb, bab,$
$bba, bbb\}$

$|L| = 14$
$|x| < 4$



0	1
00	
01	
10	
11	

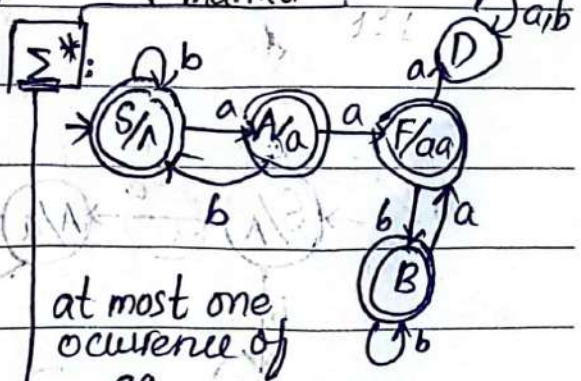
overlap is allowed more than 1 occurrence

000	x
001	
010	
011	
101	
110	
111	

2 occurrences.
← aaa

prefix	sub	suffix
aa	aa	λ
λ	aa	aa
aa	λ	aa

exactly one occurrence.
1 marked



at most one occurrence of aa

at most one ocurrence of aa

$f_n = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O\}$ // final states.

atleast one ocurrence of aa

$f_n = \{D, H, I, M\}$

exactly one ocurrence of aa

$\Sigma = \{D, I, M\}$

(null) λ ✓

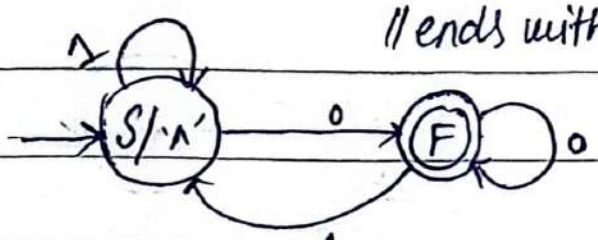
Q# In every chunk 3 (multiples of 3), there is almost one ocurrence of aa. 3, 6, 9, 12, 16 ← length of string

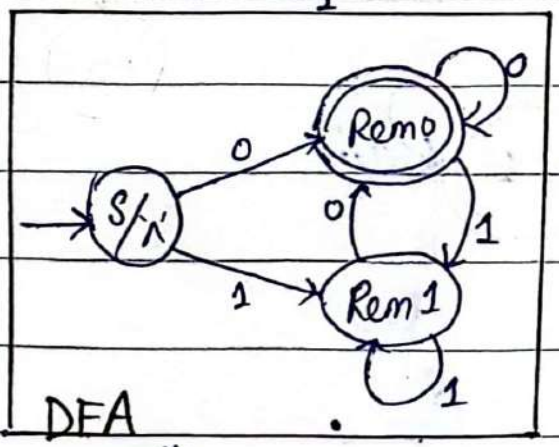
Q# $L = \{x | x \in \{0,1\}^*$; x as a decimal number divisible by 2$\}$

0, 00,

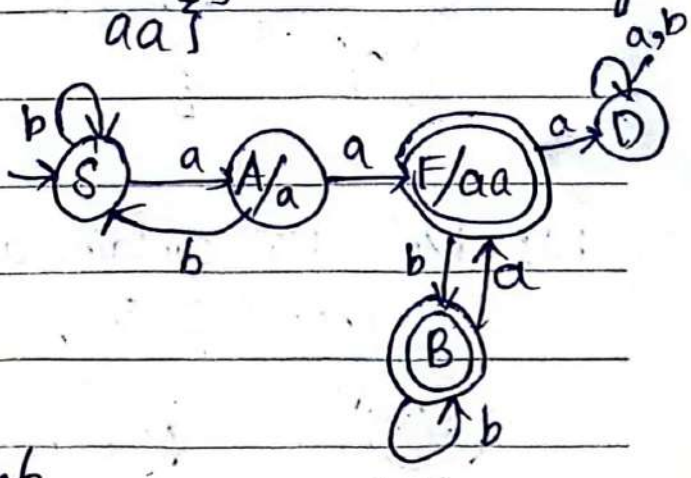
$L = \{10, 010, 100, 110, 000, \ldots\}$

// ends with 0

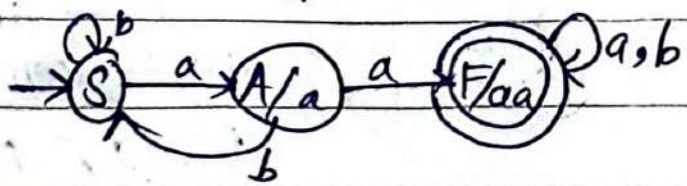




$L = \{x | x \in \Sigma^*$; x contains exactly one ocurrence of aa$\}$



$L = \{x | x \in \Sigma^*$ contains atleast one ocurrence of aa$\}$



Midterm-I 2023

Question #1:

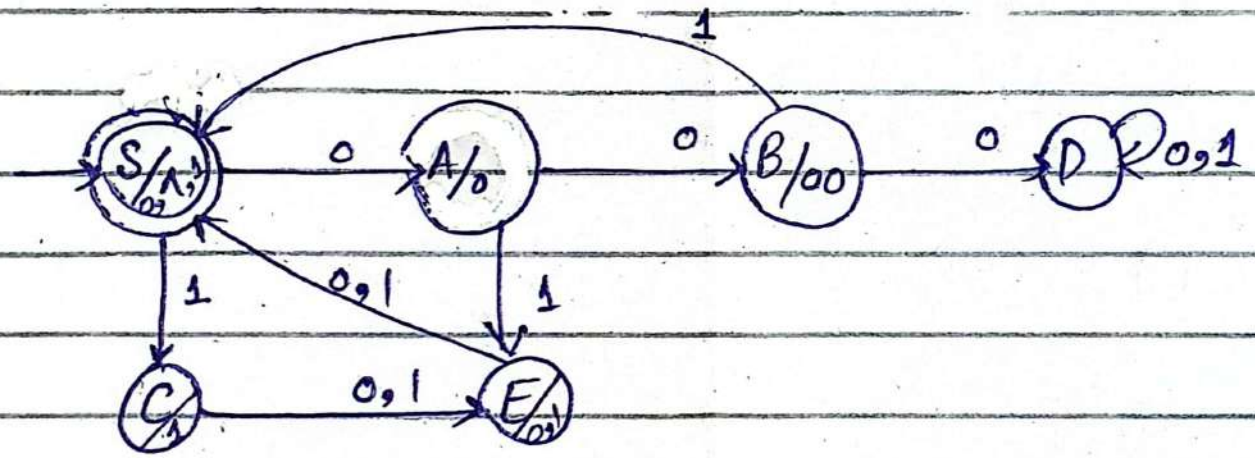
$L = \{x \mid x \in \Sigma^*$ and $|x|$ should be multiple of 3 and every three-length chunk of string contains at most two ocurenies of 0. $\}$

0,3,6

$(00)^0\ (00)^1$

$\Lambda,$

$L = \{001, 010, 011, 100, 101, 111, 001111, \ldots\}$



Closure Property on DFA:-

3/9/24

Construct a DFA

$L = \{x \mid x \in \Sigma^*$; 'aa' is not a string in x. AND x ends with 'ab' $\}$ $\Sigma = \{a, b\}$

$L = L_1 \cap L_2$ — closure property ∩, ∪, −, '

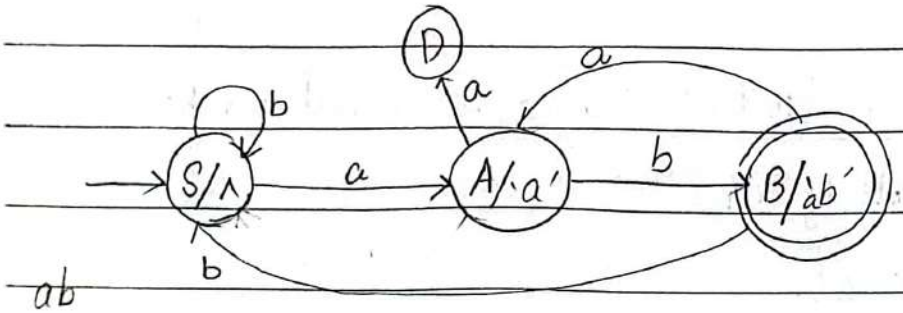
first take without not then take compliment

M_1 $L_1 = L_0'$ (L_0 don't see this aa)

L_2 M_2

$M = M_1 \cap M_2$

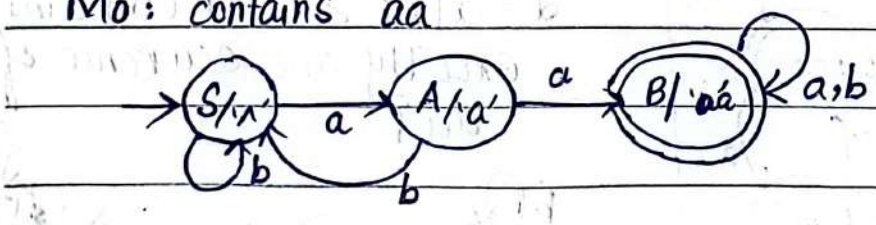
$L = \{ab, bab, abab, bbab, \ldots\}$



	Prefix	Suffix
l=0	Λ	a, aa, b, bb
l=1	a	ab, a, b, ba
l=2	ab	bb, aba, abb

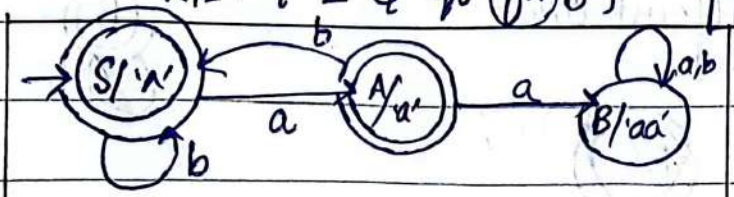
0 1
00 10
01 11
000
001 ×
010
011
100
101 bab
110
111
0000
0001
0010
0100
0101 ab
0100
0110
0111
1000
1001 ×
1010
1011
1100
1101

M_0: contains aa



$M_1 = M_0'$ NOT contains aa

M_1: $\{\Sigma\ Q\ q_0\ (fn)\ \delta\}$ $\boxed{fn_1 = Q - fn_0}$



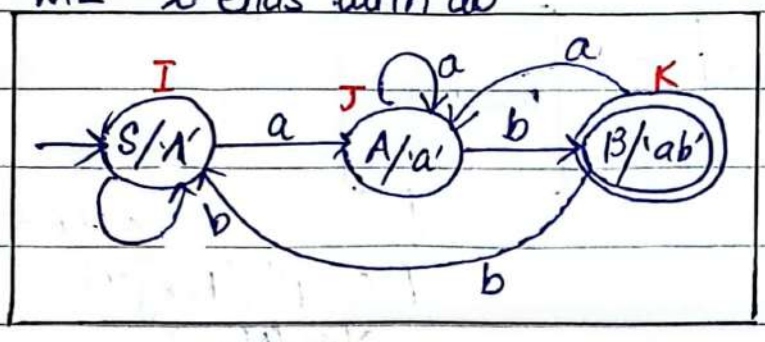
final state → non final ← will be converted.

$Q = \{S, A, F\}$

$fn = \{F\}$

$Q - fn = \{S, A\} \Rightarrow fn_1$

M2: x ends with ab



aba abb
a a b
aa ba bb
aba abb

AND

$M = M1 \cap M2$

$M1 = \{ \Sigma1, Q1, q01, fn_1, \delta1 \}$

$M2 = \{ \Sigma2, Q2, q02, fn_2, \delta2 \}$

$M = \{ \Sigma, Q, q_0, fn, \delta \}$

$\delta(p,q),s)$

$= \delta(\delta_1(p,s), \delta_2(q,s))$

$\rightarrow Q_1(Q1, Q2)$

$\Sigma1 = \Sigma2$

$q01, q02$ (SI) start state of both M1 & M2

$Q1 \times Q2$ max (9) upper bound

AND

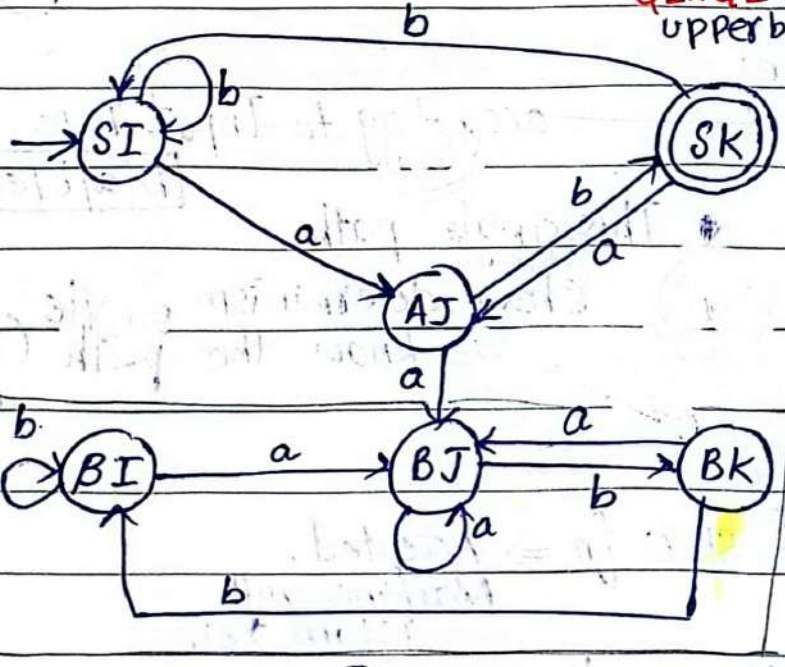
$fn_1 \times fn_2$

OR

$\{fn_1 \times Q_2\} \cup \{fn_2 \times Q_1\}$

M1 ∪ M2

eg S a A
I a J
so, SI a AJ



⇒ remove state we don't use

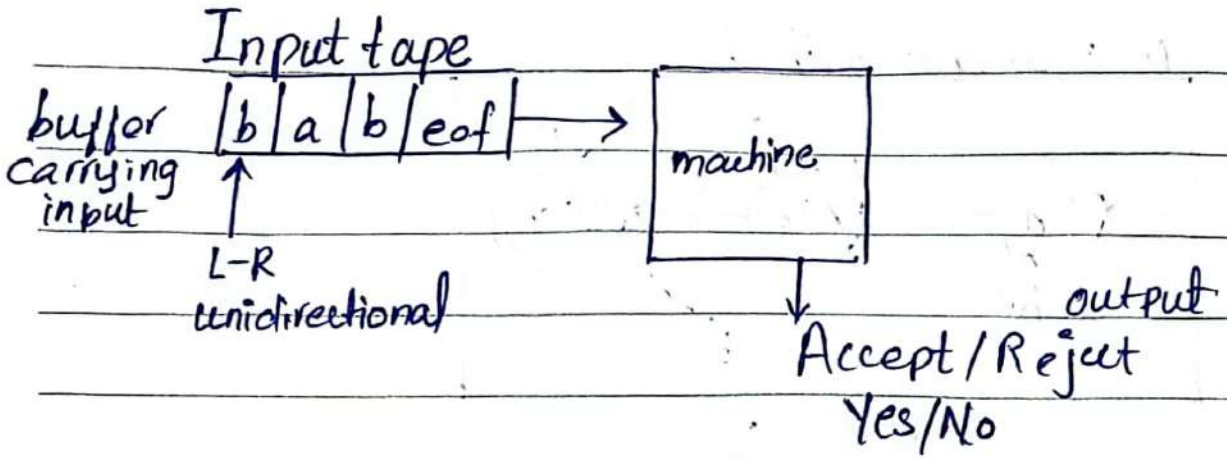
(B) can be written as deadstate ← merging

simply D a,b

replace BJ and remove BI, BK

baabb

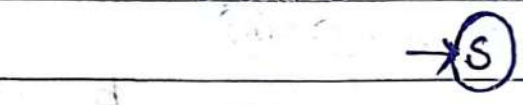
John C Martin 2.



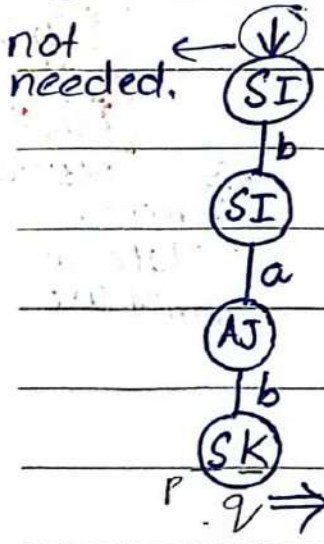
String Processing:

1) Trace Tree
2) Extended Transition Function

* if start state is not mentioned, M can start from anywhere, it will be incorrect M



(1) Trace Tree:



← according to Input Tape [b|a|b|eof]

This single path show determinism single we know the path (no arrows need)

if SK ∈ fn ⇒ Accepted. Machine will return yes.

else p, q ∉ fn ⇒ Reject.

② Extended Transition Function: δ^* we can take String input.

$q \in Q$ (set of state) $\quad \boxed{\delta^*(p, x) \to q}$ (x: string) → which means on some state p, we get an input x which is a string.

$\delta^*(SI, bab) =$

$= \delta(\delta^*(SI, \underbrace{ba}_{y}), \underbrace{b}_{6})$

$= \delta(\delta(\delta^*(SI, b), a), b)$

$= \delta(\delta(\delta(\underbrace{\delta^*(SI, \Lambda)}_{SI}, b), a), b)$

$= \delta(\underline{\delta(SI, b)}, a), b)$

$= \delta(\underline{\delta(SI, a)}, b)$ → check from machine

$= \delta(AJ, b)$

$= \boxed{SK}$ & if $\in$ fn Accepted.

Def# 2.2

Theorem for ETF

1. $\delta^*(q, \Lambda) = q$
2. $\delta^*(q, x) \Rightarrow_x$

$\delta^*(q, \underbrace{y}_{\text{string}}\underbrace{\sigma}_{\text{character}})$

$\delta(\delta^*(q, y), \sigma)$ $\sigma \in \Sigma$

5/9/24

language
↓
words
↙ creation ↘ validation

FA
↓ DFA ↓ NFA

Q# Construct DFA or NFA

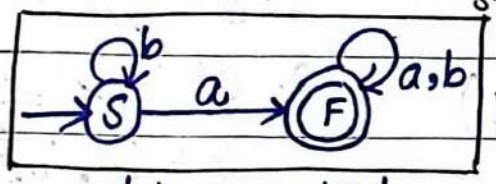
1) State machine (default)

2) Transition Table

Example: (DFA)

State Machine Method.

1)



contains 'a' or a as substring

outdegree = $|\Sigma|$ of each state

2)

columns (set of alphabets) Σ

row Q (set of states)

	a	b
-S	F	S
+F	F	F

Transition Table Method

(+ - both start & final state)

(- sign before a state means start state in transition table)

(+ for final state)

⇒ A box or cell in the table represents transitions

DFA:

All cells should be filled and every cell should consists of only 1 state

NOT empty, not more than 1

$Q \times \Sigma \to Q$

e.g. S —a→ F

$S \times a \to F$

NFA (Non-Deterministic finite Automata)

5 tuples.

1) we can skip some transitions
2) we can go to multiple states with one alphabet from a single state

common:
1) q_0
2) $fn[]$
3) $Q = [A, B, C]$
4) Σ
5) δ_{DFA} | δ_{NFA}

$Q \times \Sigma \to Q$ | $Q \times \Sigma \to P(Q)$

DFA ⊆ NFA

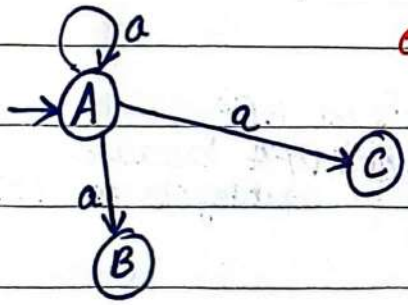
↓ powerset 2^3

$Q \subseteq P(Q)$

We can skip element. ← skip

$P(Q) = [\phi, [A], [B], [C], [A,B], [A,C], [B,C], [A,B,C]]$

Q set ← $[C]$



in powerset so it is a valid transition.

(A), a → [B, C]

dead / Trap state

* we can't specify the outbound of NFA
outgoing arrows of a state.

⇒ All DFAs are also NFAs

⇒ We design NFA to simply the design. ⇒ implementation is of deterministic machine

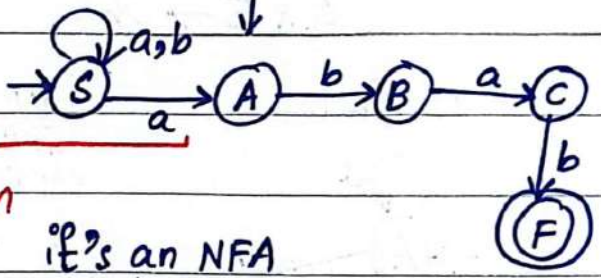
$L = \{ x \mid x \in \Sigma^* ;$ x ends with 'abab' $\}$

Σ^0 Σ^1 Σ^2

$L = \{$ abab, aabab, babab, aaabab, baabab, ababab, bbabab, ... $\}$

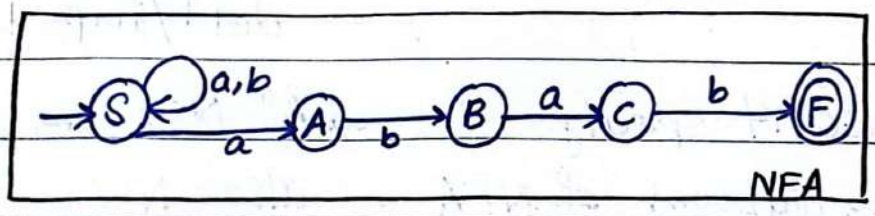
Σ^3 ..... $= \Sigma^* abab$

Σ^* machine:

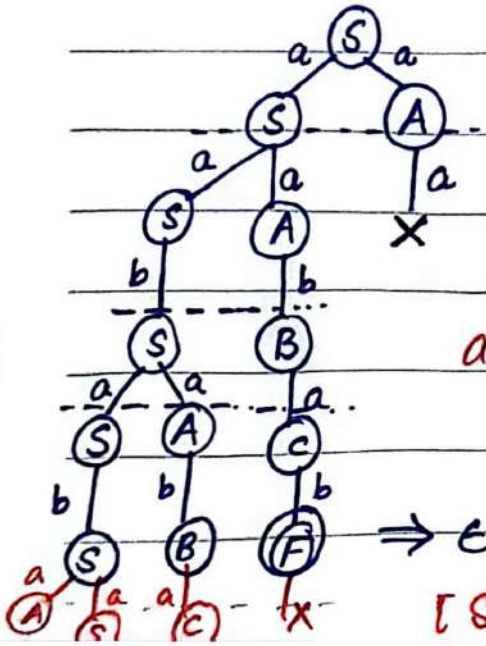


non-determinism S can go to S or A for a

it's an NFA not DFA because outdegree ≠ $|\Sigma|$



Trace Tree / Computation Tree: aababa



aababa

⇒ $\in$ fn Accepted

[S, A, C] $\notin$ fn Reject

* Every skip link is mapping to a dead state.

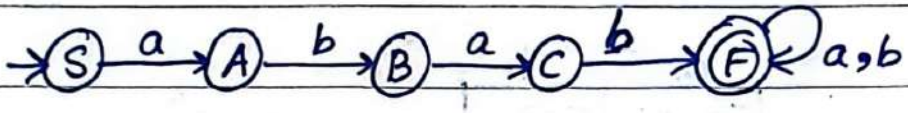
⇒ check longest path. if only one node in leaf node that belongs to fn then this string is accepted.

i.e [S, B, F*] $\in$ fn

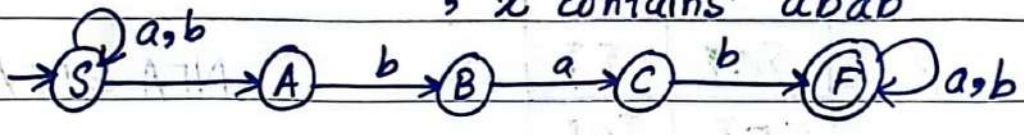
$L = \{ x / x \in \Sigma^* ;$ x starts with 'abab' $\}$

$L = \{$ abab, ababa, ababb, ababaa, ababab, ababba, ababbb, ... $\}$

$abab \in \Sigma^*$



; x contains 'abab'



* DFA will process string faster because it has a single path.

whereas in NFA, it has to detrace back if error in one path or dead end occurs.

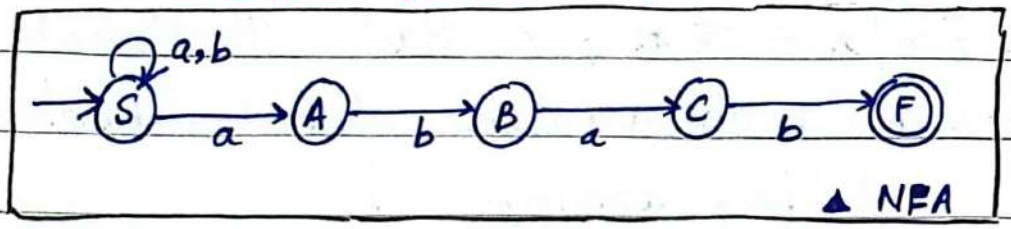
DFA:	NFA:
design tricky	design easy
processing easy	processing diff

John C. Martin.

Subset construction method

NFA to DFA conversion.

Subset Construction Method



▲ NFA

$Q \times \Sigma \rightarrow P(Q) \Rightarrow Q \times \Sigma \rightarrow Q$

NFA DFA

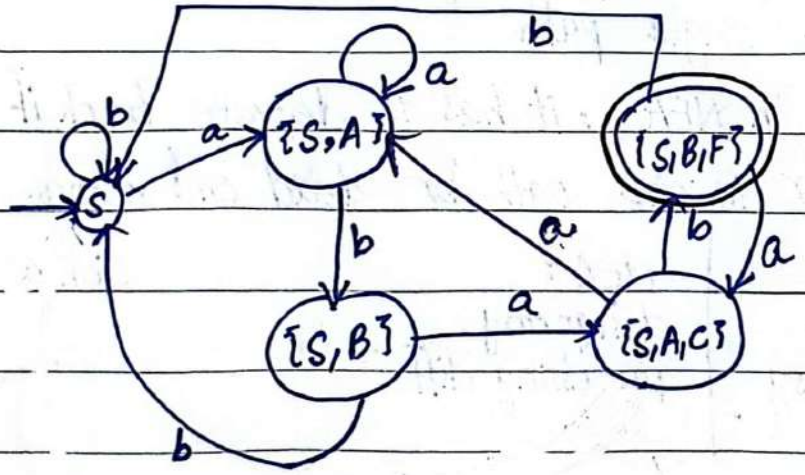
	a	b
→S	{S, A}	{S}
A	{ }	{B}
B	{C}	{ }
C	{ }	{F}
+F	{ }	{ }

SA —a→ SA

SA —b→ SB

NFA → DFA

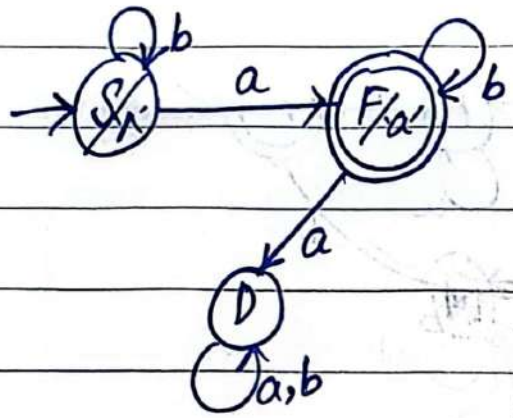
$2^Q \rightarrow 2^5 = \boxed{32}$ max states in DFA when conversion from NFA.



⇒ All (F) included states would be final states in DFA which is converted from NFA

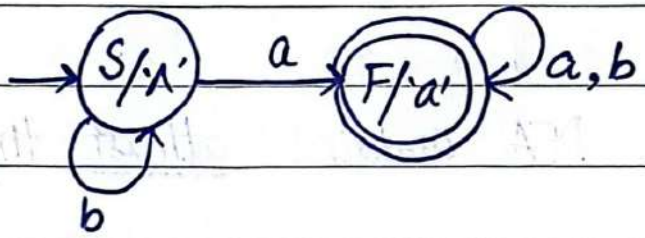
Q# Construct DFA over $\Sigma = \{a, b\}$ which accepts exactly one a

$L = \{a, ab, ba, abb, bab, bba, \dots\}$



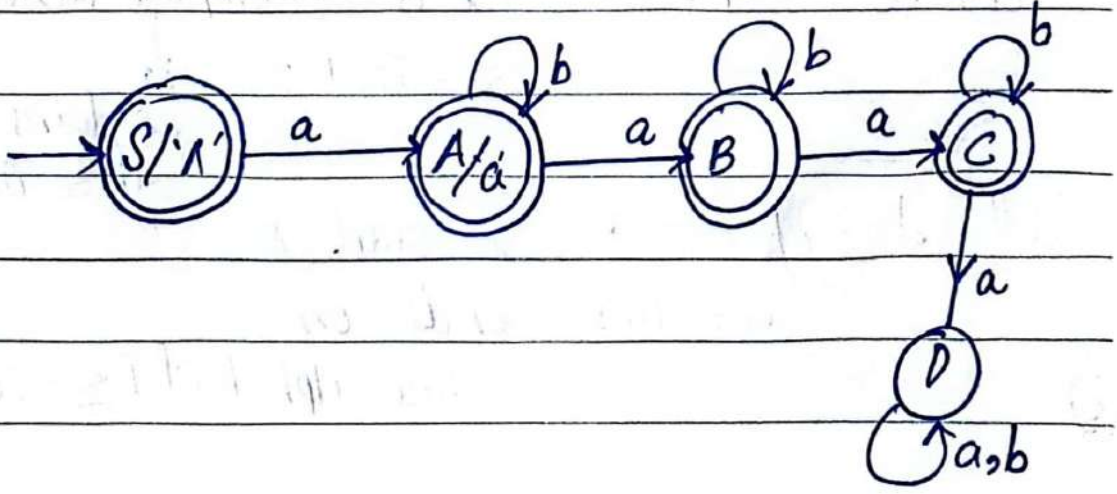
Q# Construct a DFA that contains atleast 1 a (least a)

$L = \{a, aa, ab, ba, aaa, aab, aba, \dots\}$

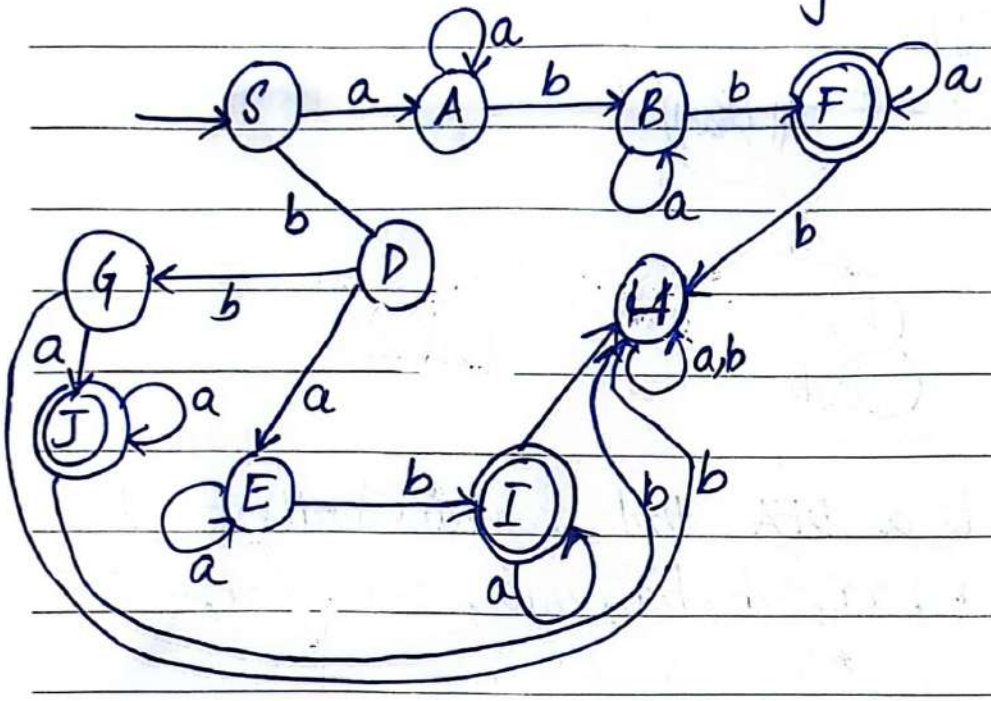


Q# Construct a DFA that contains no more than three a's $(aaa)^0\ (aaa)^1$

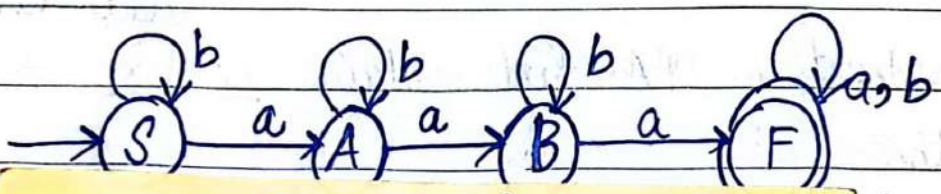
$L = \{\Lambda, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$



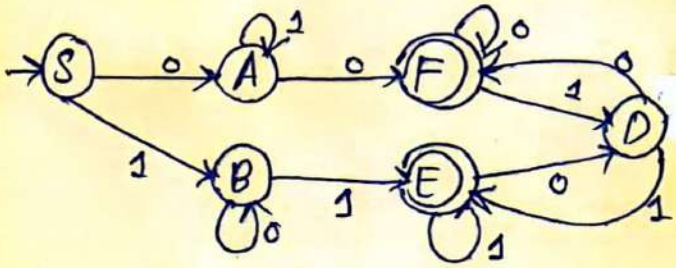
Q# Construct a DFA over $\Sigma = \{a, b\}$ which accepts all strings with atleast one a and exactly two b's



Q# construct DFA contains atleast three a's $\{a, b\}$ $(aaa)^1$ or greater



x starts and ends on same alphabet $\Sigma = \{0, 1\}$



$L = \{11, 00, 000, 010, 101, 111, 0000, \ldots\}$

...plete word in ... contains ... three 0's & 2's

...et] $\Sigma = \{0, 1\}$

John C. Martin

10/9/24

R Expression:

Regix

1) Operators (Regular)
2) Operands (Regular)
   ↳ set of alphabets $\Sigma \cup \Lambda$

{+, •, (), *, +} → Regular language (RL)

choice → +
concatenation → •
Kleene closure → *
positive star → +

validation → FA → Machines. → DFA, NFA

Expression → Regular Expression (RE)

Arithmetic Expression
e.g 1+2
operands and operators.

e.g if [R1 + R2, R1 · R2 / R1R2,
(choice) R1/R2, (), R*, R+]

1) $L = \{a, b\}$
   $RE = \{a+b\}$

2) $L = \{a\}$
   $RE = \{a\}$

3) $L = \{aa, ab, ba, bb\}$
   $RE = \{(a+b)\cdot(b+a)\}$ Ans.
   $(a+b)^2 \rightarrow a^2 + 2ab + b^2$

4) $L = \{a, b, aa, ab, ba, bb\}$
   $RE = \{((a+b)(b+a)) + a + b\}$ Ans

5) $L = \{\Lambda, a, b, aa, ab, ba, bb\}$
   $RE = \{((a+b)\cdot(a+b)) + (a+b)\}$
   $= \{((\Lambda + a + b)\cdot(a+b))\}$ Ans

OR $\Lambda a = a$, $\Lambda b = b$

CamScanner

OR

$RE = (a+b+\Lambda)^2$ ans $L = \{\Lambda, a, b, aa, ab, ba, bb\}$

0—3 all combinations cases

$(a+b+\Lambda)^3$ → it will provide you the range for lower bound and upper bound

$\{\Lambda, a, b, \ldots bbb, aaa\}$

0 till N combinations

$(a+b+\Lambda)^N$

$(a+b)^N$ only N combinations

6) $L = \{\Lambda, a, b, aa, ab, ba, bbb, \ldots\}$

$= \boxed{(a+b)^*}$ infinite combinations $\Rightarrow (a+b)^{(2)}$ two size only.

$\{aa, bb, ab, ba\}$

7) $L = \{a, b, aa, ab, ba, bbb, \ldots\}$

$= \boxed{(a+b)+}$ $\Rightarrow (a^2+b^2)$

$\{aa, bb\}$

8) $L = \{x \mid x \in \Sigma^*$; x ends with 'aba'$\}$

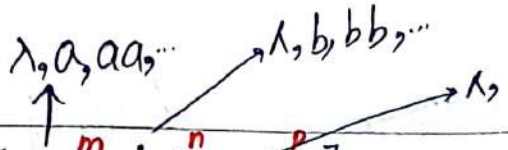
$L = \{aba, aaba, baba, aaba, abaaba, \ldots\}$

$RE = \boxed{(a+b)^* \cdot a \cdot b \cdot a}$

9) $L = \{x \mid x \in \Sigma^*$; x contains 'aba'$\}$

$L = \{aba, aaba, baba, abaa, abab, \ldots\}$

$RE = \boxed{(a+b)^* \cdot a \cdot b \cdot a \cdot (a+b)^*}$



$L = \{ a^{*m} . b^{*n} . c^{*p} \}$

$m, n, p \geq 0$
they should not mandatory be same it can not be and can be

$L = \{ \lambda, abc, aa, ab, ac, bb, bc, cc \ldots \}$

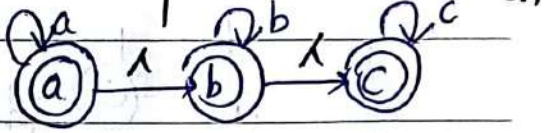
ba (cannot write) like these

NFA-λ

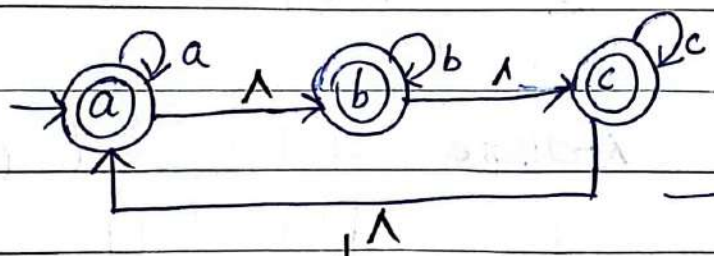
NFA-λ:

$\delta_{NFA-\lambda} \quad Q \times (\Sigma \cup \lambda) \rightarrow P(Q)$

we can use it for concatenation



$L = \{ \underbrace{a^*}_{L1} . \underbrace{b^*}_{L2} , \underbrace{c^*}_{L3} \}$

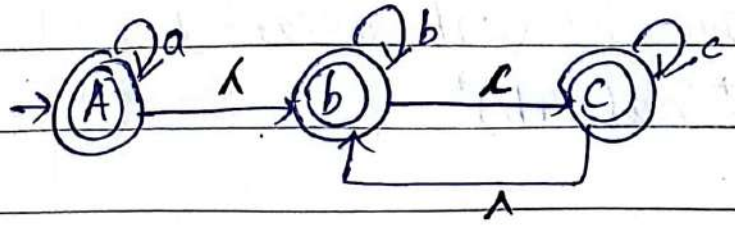


$L = (a^* . b^* . c^*)^*$

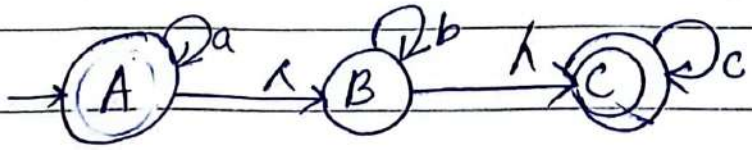
$\Downarrow$

$(a+b+c)^*$

we can use it for loop purpose



$a^* . (b^* . c^*)^*$



Null closure

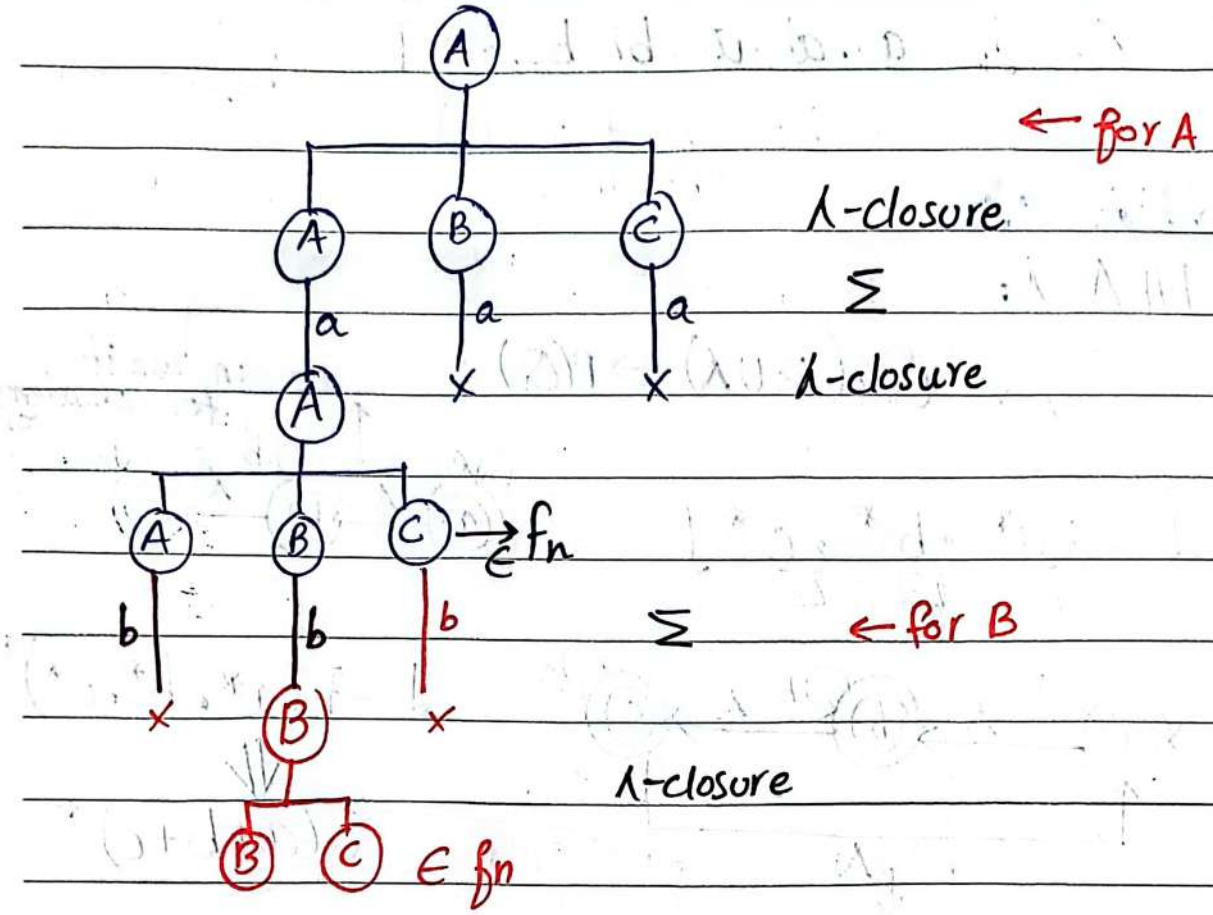
λ will get accepted it will go without reading

$\lambda(A) = \{ A, B, C \}$

$\lambda(B) = \{ B, C \}$ (Null closure) include state

if $\{ C \}$ (Null transition) donot include

$\Lambda(C) = \{C\}$ Null Closure Λtransition = empty. of it



Q#: $L = \{x \mid x \in \Sigma^*$; x contains aa$\}$ $\Sigma = \{a, b\}$

$L = \{aa, aaa, baa, aab, aaaa, \ldots\}$

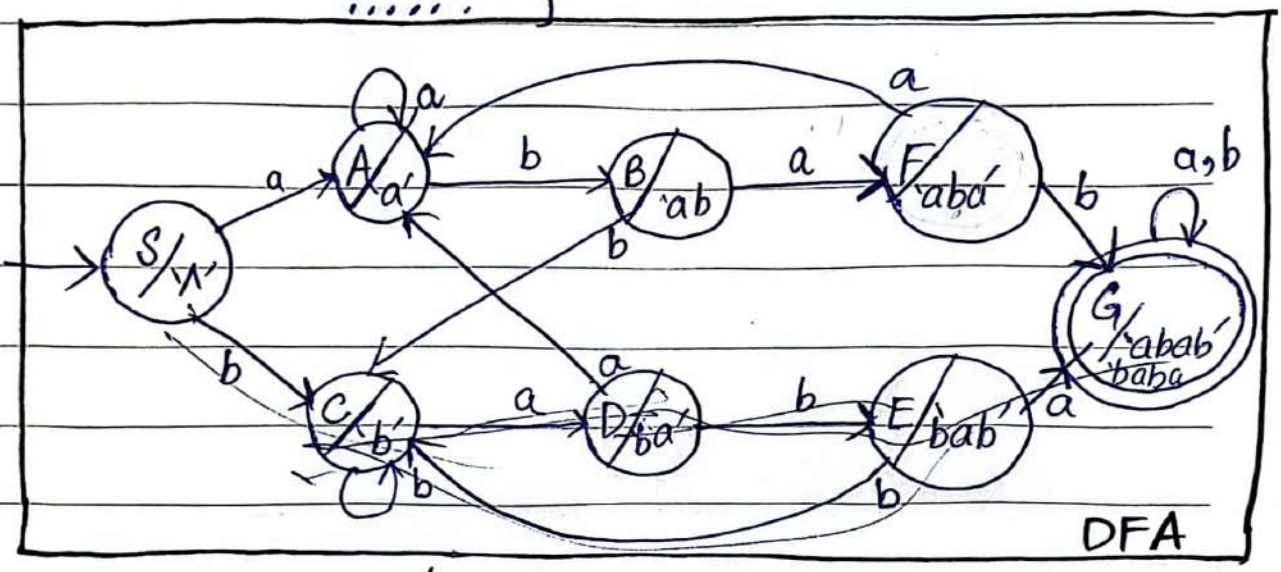
$(a+b)^* aa(a+b)^*$

Λ also added in it

C3: 22L-6946 Fatima Imran Quiz #1

$L = \{x \mid x \in \Sigma^* ;$ x contains aba & bab as a substring $\}$ $\Sigma = \{a, b\}$

$L = \{$ abab , baba , aabab , babab , ababa , bbaba, ... $\}$



a → a aa
b ba

b → b bb

aa
bb

C bb
D baa

aa
baa

aB → b abb

bb
abb

babb

bb
abb
babb

abab

aa
baa
abaa
abaabaabab

abbaba

babaa
babbaba

abaaba ×
bbbb ×
ababba

babba

abaabbab

12 Sep 2024 RE and state elimination method

1) $L = \{x \mid x \in \Sigma^* ; |x| \leq 50\}$ $\Sigma = \{a, b\}$

$L = \{\lambda, a, b, aa, ab, ba, bb, \ldots\}$ till 50 combinations

$\Sigma^0 \cup \ldots \cup \Sigma^{50}$.

0 —— 50.

$\boxed{(a+b+\lambda)^{50}}$ Answer

2) $L = \{x \mid x \in \Sigma^* ;$ x contains 'aba' & 'bab' as substring$\}$ $\Sigma = \{a, b\}$

$L = \{abab, baba, aabab, ababa, \ldots\}$

$RE = [(a+b)^* abab (a+b)^* + (a+b)^* baba (a+b)^*]$

6, 7++ length $+ [(a+b)^* aba (a+b)^* bab (a+b)^*$

$+ (a+b)^* bab (a+b)^* aba (a+b)^*]$

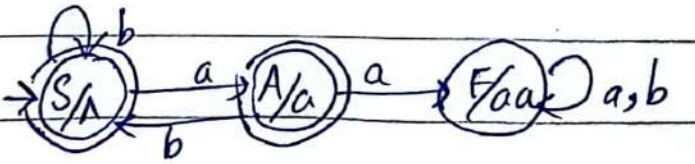
$L = \{x \mid x \in \Sigma^*$; x contains doesnot contains 'aa'$\}$ DFA

$\{(a+b)^* (aa) (a+b)^*\}$ contains aa.

$L = \{\lambda, a, b, ab, ba, bb, aba, \ldots\}$

~~$(a+b)$~~ ~~$a^* + b^*$~~

doesnot accept 'aa'

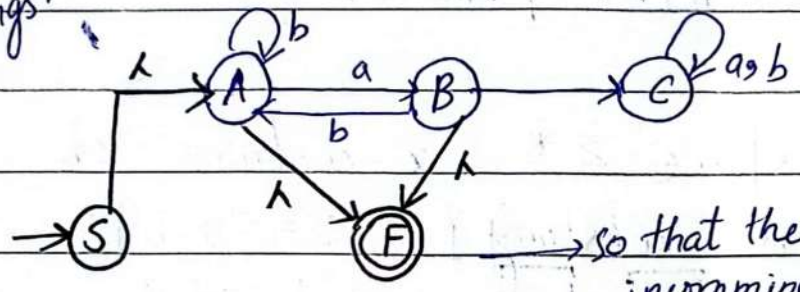


conversion

DFA ⟶ RE

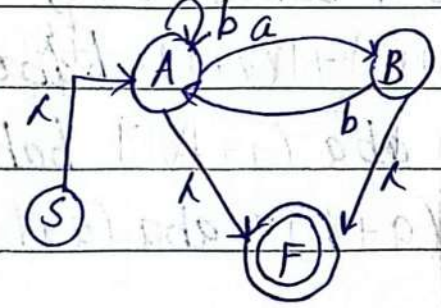
Step1: GNFA (Generalized non-deterministic finite automata)

can write alphabet combination, strings.

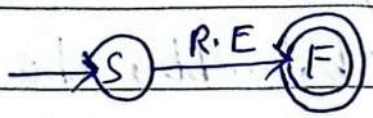


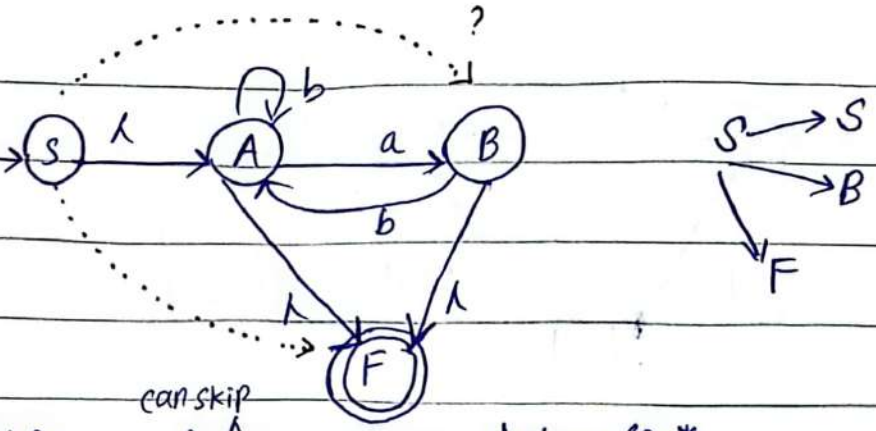
⟶ so that there is no incomming arrow in (S) and outgoing arrow in (F)

Step2: Delete dead state



Step3: Delete all nodes step by step except SBF one by one in ascending order.

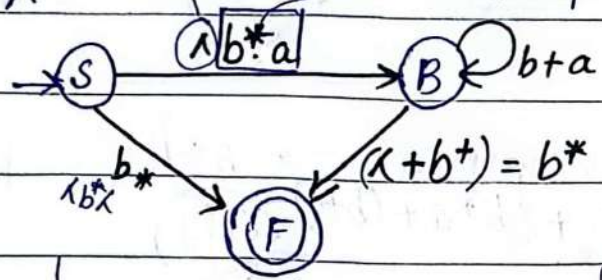




can skip λ

Delete 'A'

b loop so *



	Direct Path	Indirect Path
no possibility S — S	×	S → A → A
S — B	×	S → A → B
S — F	×	S → A → F

Source node — deleting node — dest node.

loop is included.

~~B — S~~	×	
B — F	λ	+ B–A–A–F
B — B	×	B –b→ A –b*→ A –a→ B

b b* λ

discard

b b* (λ)

b+ Ans.

both paths direct or indirect write by (+)

e.g

B – F

(λ + b+)

A – A, b b*a

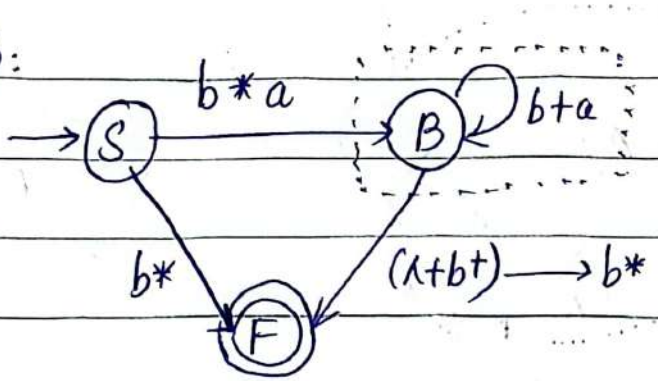
b+a Ans.

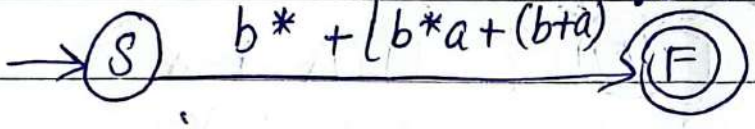
* $(b^+ + \lambda) = b^*$

$b \cdot b^* = b^+$

$(a^*)^* = a^*$

delete B:





$$R.E = b^* + [b^*a + (b+a)^* \cdot b^*]$$

$b^0 = \lambda$

(optimize minimize) ↑ m DFA

State elimination method

DFA → RE

Subset construction theorem.

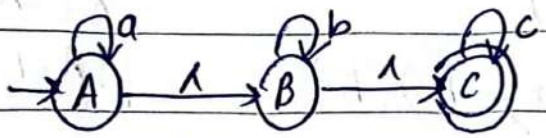
Kleene's theorem.

NFA ← NFA − λ

↑ elimination method

19/9/24

NFA = $\Sigma = \{a, b, c, \lambda\}$



▲ it is NFA for $\Sigma = \{a, b, c, \lambda\}$

▲ it is NFA-λ for $\Sigma = \{a, b, c\}$

NFA-λ: if λ is not a part of language as alphabet e.g $\{a, b, c\}$

$\delta(Q \times [\Sigma \cup \lambda]) \rightarrow P(Q)$

abb:

String processing:

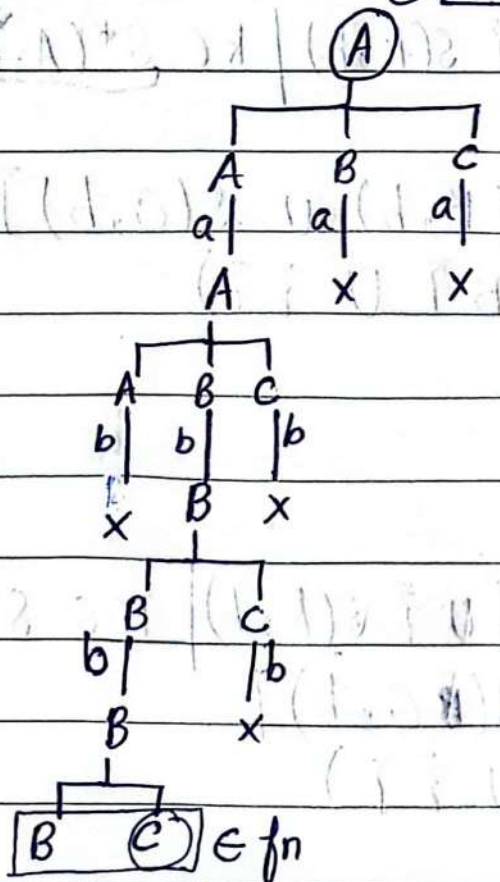
1) Trace Tree
2) Extended Transition function.

$\lambda(A) = \{A, B, C\}$

$\lambda(B) = \{B, C\}$

$\lambda(C) = \{C\}$

① **Trace Tree:**



Extended transition function:

DFA $\delta^*(A, \lambda) = A$

1) $\delta^*(A, \lambda)$
$= \lambda([A])$
$= \boxed{[A, B, C]}$

$\delta^*(A, abb)$
$\delta(\delta(\delta(\delta^*(A, \lambda), a), b), b)$

(NFA) because states more than one. ← union

2) $\delta^*(A, a) = \lambda\left(\bigcup [\delta(K, a) \mid K \in \underbrace{\delta^*(A, \lambda)}_{\text{previous step.}}]\right)$ $[A, B, C]$

$= \lambda(\delta(A, a) \cup \delta(B, a) \cup \delta(C, a))$
$= \lambda([A] \cup [\,] \cup [\,])$
$= \lambda([A])$
$= \boxed{[A, B, C]}$

3) $\delta^*(A, ab) = \lambda\left(\bigcup [\delta(K, \underbrace{b}_{\text{last ch}}) \mid K \in \underbrace{\delta^*(A, a)}_{\text{ans of last}}]\right)$

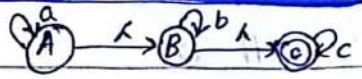
$= \lambda(\delta(A, b) \cup \delta(B, b) \cup \delta(C, b)]$
$= \lambda([\,] \cup [B] \cup [\,])$
$= \lambda([B])$
$= \boxed{[B, C]}$

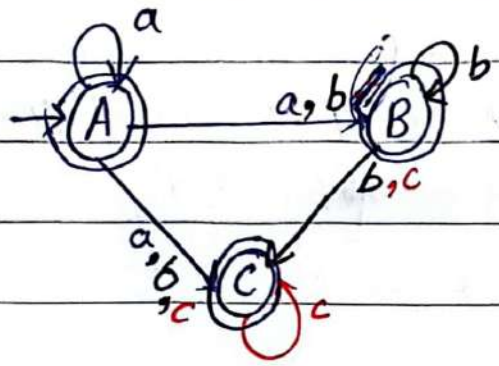
4) $\delta^*(A, abb) = \lambda[\bigcup [\delta(K, b) \mid K \in \delta^*(A, ab)]$
$= \lambda(\delta(B, b) \cup \delta(C, b)]$
$= \lambda([B] \cup [\,])$
$= \lambda([B])$
$= \boxed{[B, C] \in [n]}$ so abb is part of language.

C in last so accepted.

NFA λ – NFA

λ elimination Method : NFA-λ → NFA

NFA-λ = 



DFA $Q \times \Sigma \rightarrow Q$ ↑ change in states.

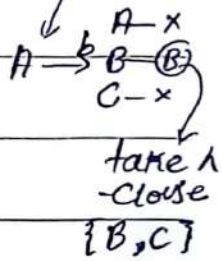
NFA $Q \times \Sigma \rightarrow P(Q)$ ↑ no change in states

NFA-λ $Q \times (\Sigma \cup \lambda) \rightarrow P(Q)$

just change in no of transitions.

1)

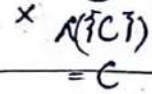
State	λ	Σ	λ
A	A, B, C	a	{A, B, C}
A	B, C	b	{B, C}
A	C	c	{C}



* (λ) accepted in NFA-λ

2)

State	λ	Σ	λ
B	{B, C}	a	{ }
B	{B, C}	b	{B, C}
B	{B, C}	c	{C} (B–C)



3)

C	{C}	a	{ }
C	{C}	b	{ }
C	{C}	c	{C} (C–C)

4) λ(State) = contains final state

make → state the final state

λ(A) = {A, B, (C)} make A final

λ(B) = {B, (C)} make B final.

OR

just make start state final

if we trace back λ (A) –λ→ B –λ→ (C)

incoming λ

RE — NFA - λ

→ make machines for small small REs

Kleene's Theorem :-

R.E = $(a+b)^* a$ 'ends with a'

R_1 + R_2

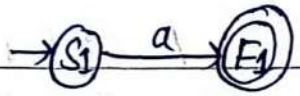
made up of small small Regular exp.

$R_3 = R_1 + R_2$

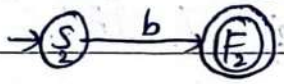
$R_4 = R_3^*$

$R_5 = R_4 \cdot a$

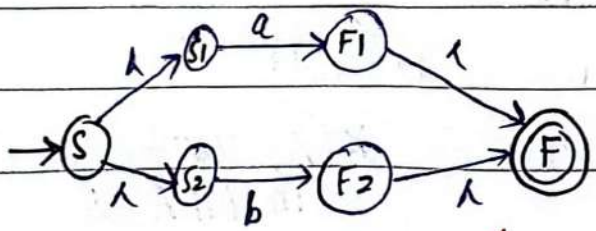
M1:



M2:

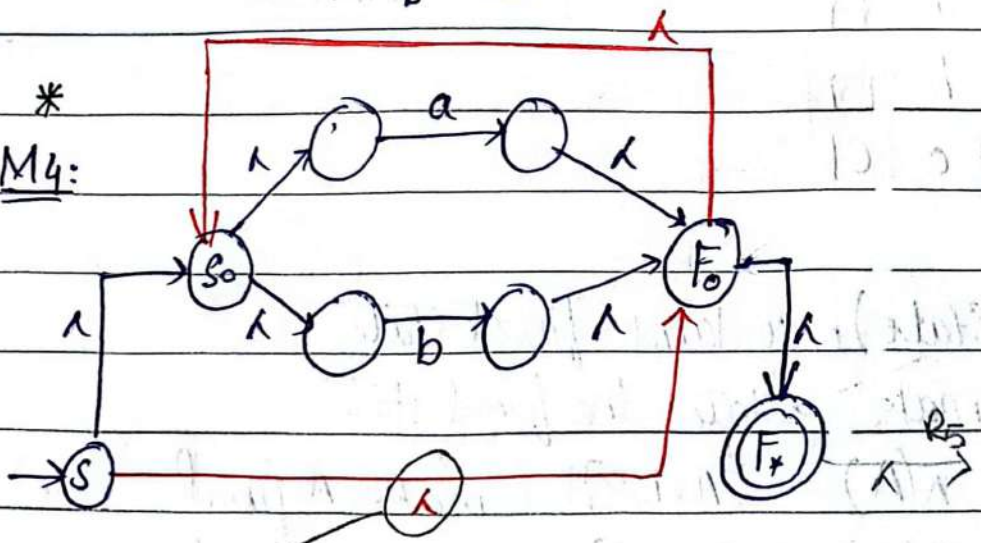


M3:



for choice (+) make new start and final and join by NULL

* M4:



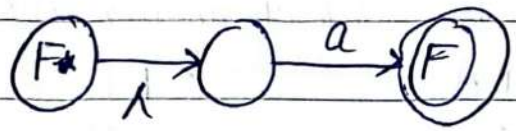
if we skip this link, λ won't be accepted (a+b)* machine

old start to old final

newstart to old final.

M5:

" " "



Total 10 states. NFA-λ

10 ↓ NFA

$2^{10} = 1024$ ↓ DFA

↓

we can minimize DFA and remove extra states.

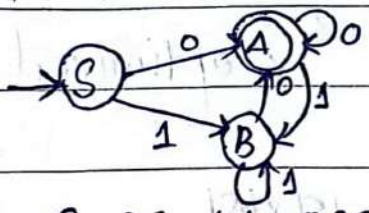
M1

$L = \{x \mid x \in \Sigma^*$; x as number & sum of alphabet is divisible by 2 $\}$

M2

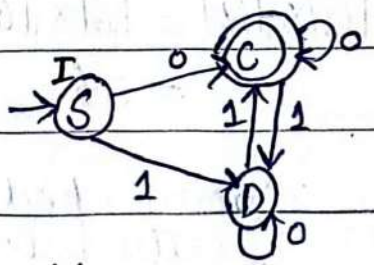
$\Sigma = \{0, 1\}$

$L_1 = \{0, 10, 00, 000, 010, 100, \cdots\}$



M1

$L_2 = \{\lambda, 0, 00, 11, 000, 011, 101, 110, 0000, \cdots\}$



M2

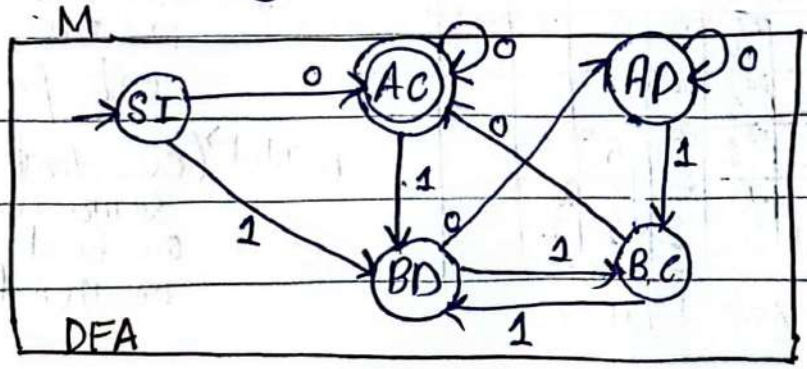
A → 0 → A B → 0 → A

A → 1 → B B → 1 → B

C → 0 → C D → 0 → D

C → 1 → D D → 1 → C

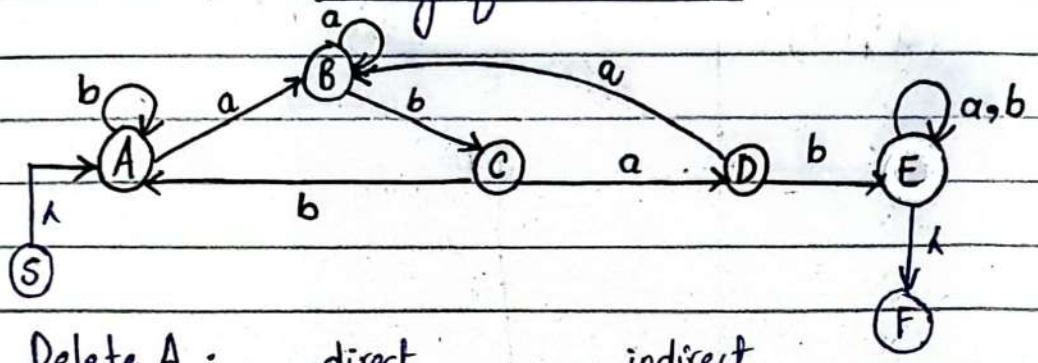
M



11

22L-6946
Fatima Imran

Homework
Theory of Automata

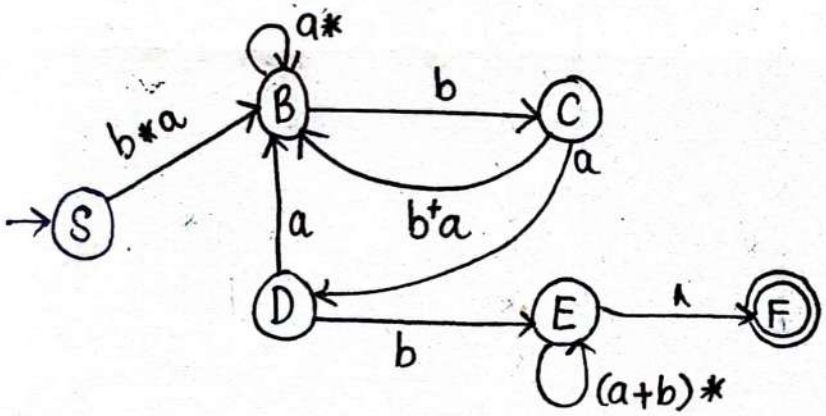


Delete A :

	direct	indirect	
S – B	*	λb*a	= b*a
S – C	x	x	
S – D	x	x	
S – E	x	x	
S – F	x	x	
B – B	a*	x	= a*
B – C	b	x	= b
B – D	x	x	
B – E	x	x	
B – F	x	x	
C – B	x	bb*a	= b+a
C – C	x	x	
C – D	a	x	= a
C – E	x	x	
C – F	x	x	
D – B	a	x	= a
D – C	x	x	
D – D	x	x	

CamScanner

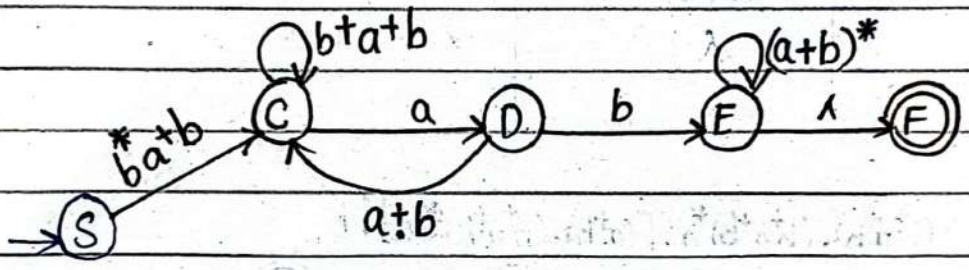
	direct	indirect	
D–E	b	x	= b
D–F	x	x	
E–B	x	x	
E–C	x	x	
E–D	x	x	
E–E	$(a+b)^*$	x	$= (a+b)^*$
E–F	$\wedge$	x	
F–B	x	x	no outgoing arrows.
F–C	x	x	
F–D	x	x	
F–E	x	x	
F–F	x	x	



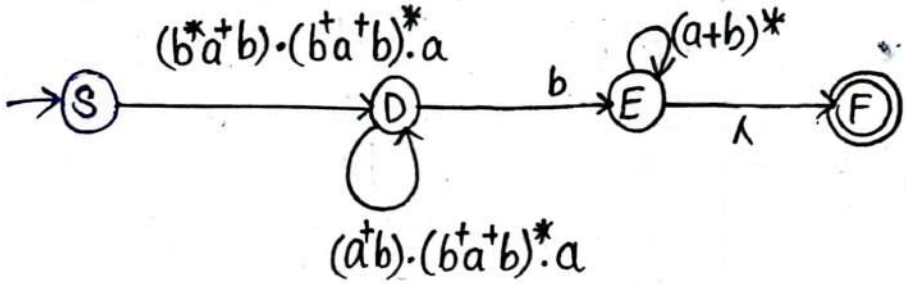
Delete B:

	direct	indirect	
S–C	x	b^*aa^*b	$= b^*a+b$
S–D	x	x	
S–E	x	x	
S–F	x	x	
C–C	x	b^+aa^*b	$= b^+a^+b$
C–D	a	x	$= a$
C–E	x	x	
C–F	x	x	

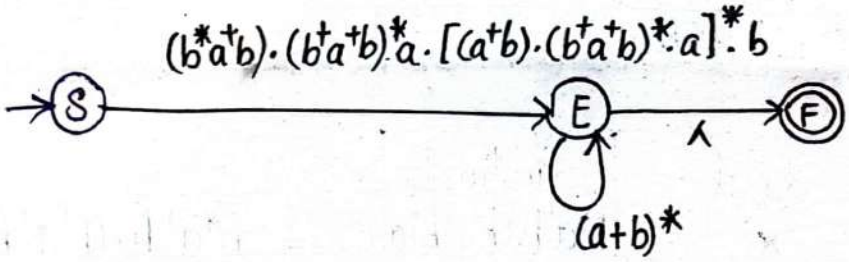
	direct	indirect	
D — C	x	aa*b	= a+b
D — D	x	x	
D — E	b	x	= b
D — F	x	x	
E — C	x	x	
E — D	x	x	
E — E	(a+b)*	x	= (a+b)*
E — F	λ	x	= λ



Delete C:	direct	indirect	
S — D	x	$b^*a^+b(b^+a^+b)^*a$	$= b^*a^+b\cdot(b^+a^+b)^*\cdot a$
S — E	x	x	
S — F	x	x	
D — D	x	$(a^+b)(b^+a^+b)^*a$	$= a^+b(b^+a^+b)^*\cdot a$
D — E	b	x	= b
D — F	x	x	
E — D	x	x	
E — E	(a+b)*	x	= (a+b)*
E — F	λ	x	



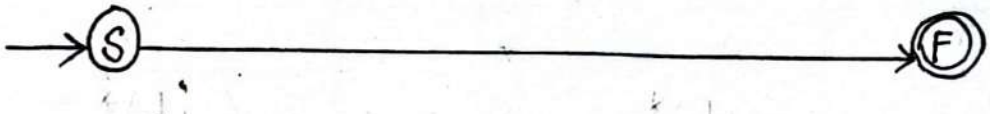
Delete D:	direct	indirect
S − E	×	$(b^*a^+b)\cdot(b^+a^+b)\cdot a\cdot[(a^+b)\cdot(b^+a^+b)^*a]^*\cdot b$
S − F	×	×
E − E	$(a+b)^*$	×
E − F	λ	×



Delete E:

S − F $\quad (b^*a^+b)\cdot(b^+a^+b)^*\cdot a\,[(a^+b)\cdot(b^+a^+b)^*.a]^*.b\cdot[(a+b)^*]^*\lambda$

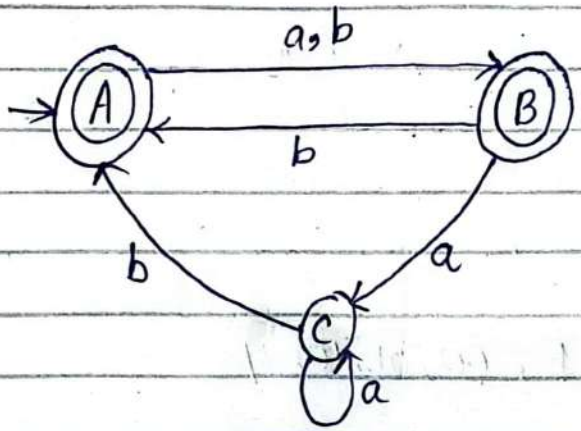
$(b^*a^+b)\cdot(b^+a^+b)^*a\cdot[(a^+b)\cdot(b^+a^+b)^*.a\,]^*b\cdot\underline{((a+b)^*)^*} = (a+b)^*$



$R.E = (b^*a^+b)\cdot(b^+a^+b)^*a\cdot[(a^+b)(b^+a^+b)^*.a]^*b\cdot(a+b)^*$

Answer

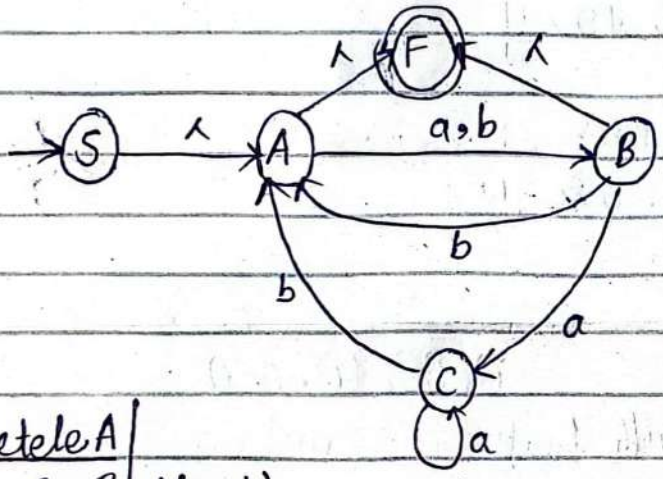
DFA to RE



delete B

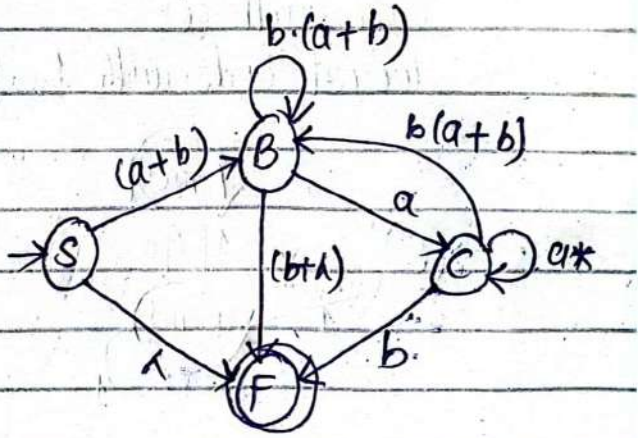
S-C	$(a+b)[b(a+b)]^*a$
S-F	λ
C-C	$a^*+[b(a+b)$
C-F	$[b(a+b)]^*a]$

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

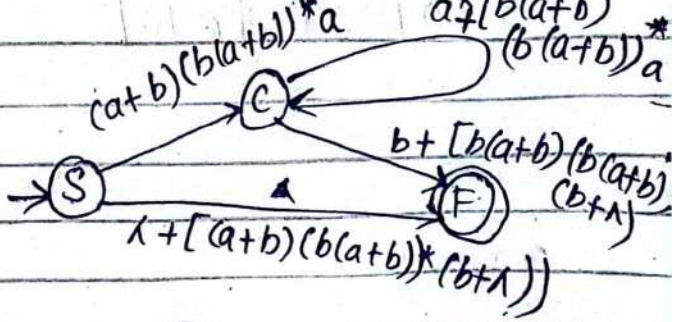


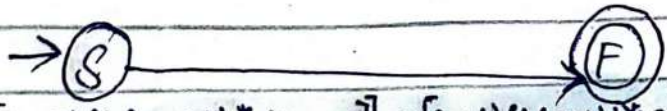
delete A

S-B	$\lambda(a+b)$
S-C	X
S-F	λ
B-B	$b\cdot(a+b)$
B-C	X a
B-F	$\lambda+(b+\lambda)$
C-C	a^*
C-B	x $b(a+b)$
C-F	x $b\lambda$



delete C:

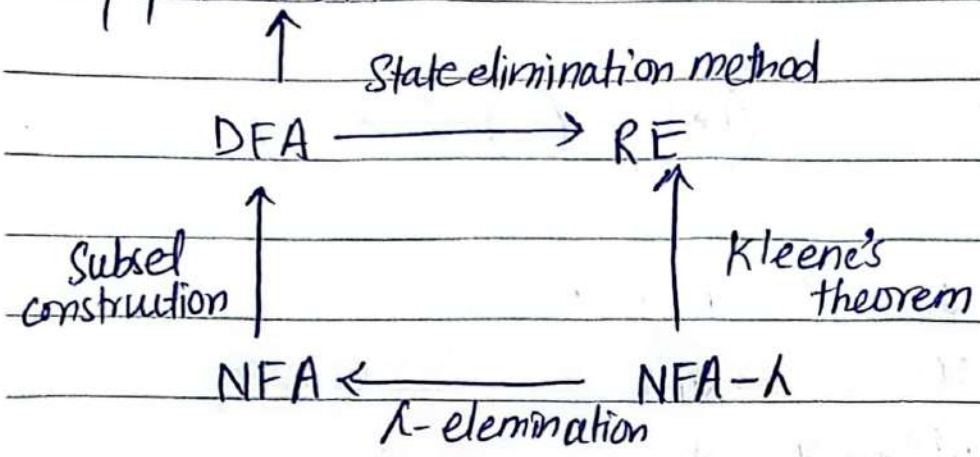


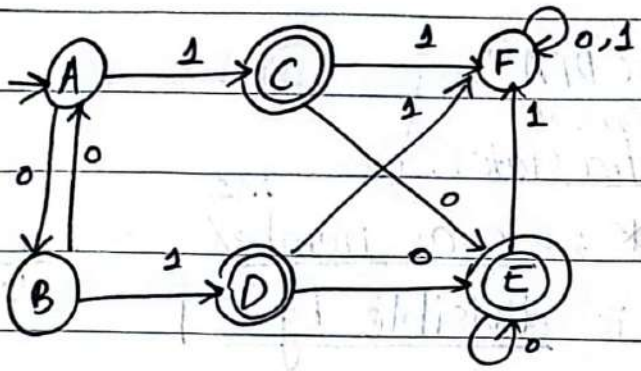


$$[\lambda+[(a+b)(b(a+b))^*(b+\lambda)]+[(a+b)(b(a+b))^*a]\cdot[a^*+b(a+b)(b(a+b))^*a]^*$$
$$[b+b(a+b)\cdot(b(a+b))^*(b+\lambda)]$$

DFA → mDFA

26/9/24

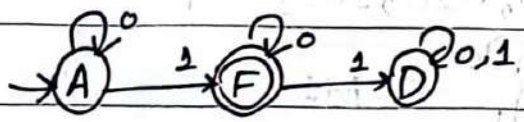




← sub optimal

$fn = \{G, D, E\}$ (final)

$fn' = \{A, B, F\}$ (non final)



← optimized

Step1: Grid of size $|Q| \times |Q|$

$Q = 6$ $\quad |Q| \times |Q| = |6| \times |6| = 36$

	A	B	C	D	E	F
A	AA	AB				
B	BA					
C	CA 1	CB 1	CD	CD	CE	CE
D	DA 1	DB 1	DC			
E	EA 1	EB 1	EC	ED		
F	FA 2	FB 2	FC 1	FD 1	FE 1	

→ donot check states belonging to same set fn or fn'

marked → (cuz don't have same behavior) one final one non-final.

▲ 1st iteration

2nd iteration :-

BA
- 0 → A B ⇒ (BA)
- 1 → D C ⇒ (CD)

source boxes.

destination boxes.

if any one of dest box is marked then mark source too.

↓ both empty so can't fill

GD
- 0 → E E ⇒ (EE)
- 1 → F F ⇒ (FF)

discarded boxes (diagonal)

EC
- 0 → E E ⇒ (EE)
- 1 → F F ⇒ (FF)

ED
- 0 → E E ⇒ (EE)
- 1 → F F ⇒ (FF)

FA
- 0 → F B ⇒ (FB)
- 1 → F C ⇒ (FC) marked (1)

FB
- 0 → F A ⇒ (FA) marked (2)
- 1 → F D ⇒ (FD) marked (1)

3rd iteration: (these were marked 2.)

check FA (2) FB (2)

in which destinations. then we have to mark [3]

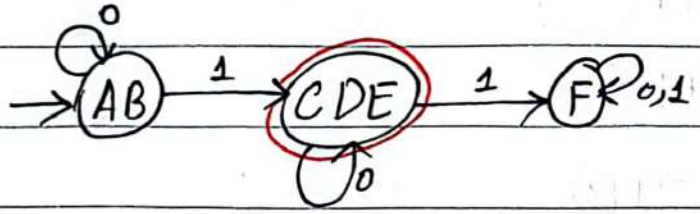
⇒ when previous or next table (current). no change then stop (work non stop till both iterations have same result)

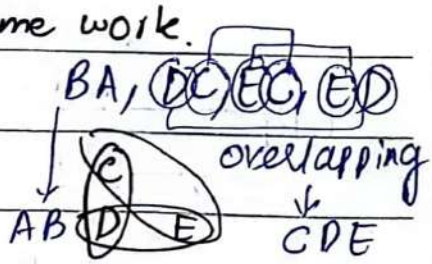
3rd iteration:

BA → BA, DC	non marked and same state as 2nd iteration so stop here.
DC → EE, FF	
EC → EE, FF	
ED → EE, FF	

→ merge states that are [empty] → assume they have same type of transitions

means they have same work.





formed a cycle so merge all three or more together.

CD DE CE

overlap and form cycle.

This algo will make sure empty boxes have same behaviour

final = C, D, E

if in diff states

e.g CA DB

make both final

Pumping Lamma for RL

pg#63
John C Martin

1/10/24

& $n! = m$

$L = \{ a^n b^m ; n, m \geq 0 \}$ & $n! = m$ & $n, m \geq 2$

n	m	words
0	0	$\wedge$
0	1	b
1	0	a
1	1	ab
2	0	aa
0	2	bb

$R.E = \{ a^* b^* \}$

$b = a^0 b^1$
$a = a^1 b^0$
$bb = a^0 b^2$
$aa = a^2 b^0$
$abb = a^1 b^2$
$aab = a^2 b^1$

n	m	
0	1	a
1	0	b
2	0	aa
0	2	bb

no null.

$a^* b^*$ can be same.

RL RE/FA

~RL?
non-regular language

$\Rightarrow$ Pumping Lamma is a property for L

assume RL → PP
regular language has pumping property

same

$L = \{ a^n b^n ; n \geq 0 \}$

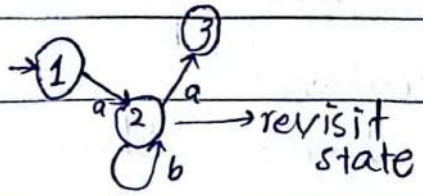
RL → PP (Pumping property)

1) $\forall x \in L$ when $|x| \geq n$, no of states in your FA
2) $\exists$ uvw ; such that
   $x = uvw$ (u → prefix, v → substring, w → suffix)
3) $\forall i \geq 0$; $uv^i w$ is also in L.

looping element

e.g aba valid
ab invalid for 3 states

$L = \{ ab^* a \}$
smallest word = aa



considering property ①

$n=3$

can't pick aa due to $|x| \geq n$

$L_{new} = \{abba, abbba, \dots\}$

$x \rightarrow u = a$

$v = b$

$w = a$

$i \geq 0$ $uv^i w$

a b a

$i=0$ aa

pump down property (if new word it's part of L)

$i=2$ abba

pump up $\rightarrow i > 2$

$i=1$ aba

no pumping scenario.

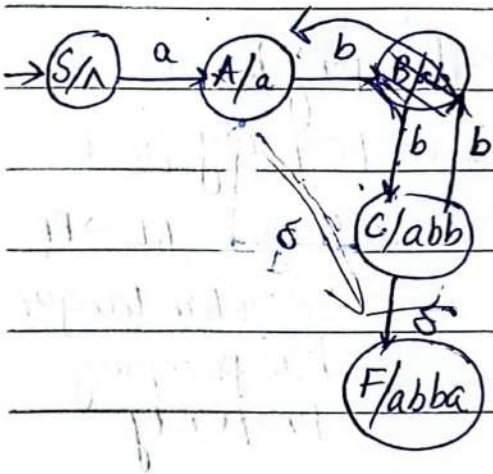
$L = a(bb)^* a$

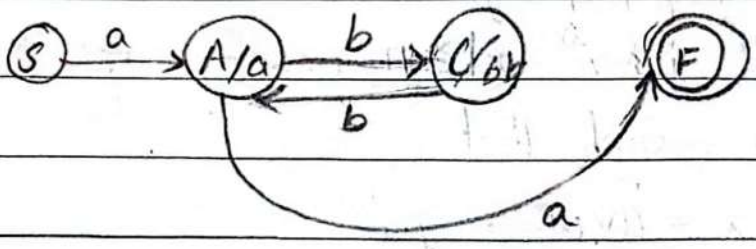
abba

abbbba

$L_{orig} = \{aa, aba, abba, \dots\}$

ab^*a



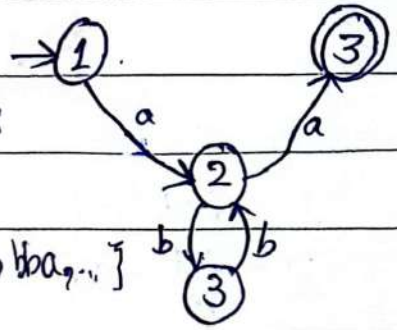


$L = a(bb)^* a$

even combinations of b

Smallest word

$L_{new} = \{abba, abbbba, \dots\}$



$x = abba$

$u = a$ $\quad i=0 \quad x = aa$

$v = bb$ $\quad i=2 \quad x = abbbba$

$w = a$ $\quad i=1 \quad x = abba$

$x = abba$

$u = ab$ $\quad aba \quad i=0$

$v = b$ $\quad \downarrow$ can't be in language $(abb)^*a$

$w = a$

→ Pumping Lamma is not used to prove Regular Language. $\quad RL \rightarrow PP$

REA $\quad$ R.E

it is used to disprove or tell about non-regular language.

$\sim PP \rightarrow \sim RL$ contrapositive

PP not exit $\quad$ not a RL

$\sim PP \rightarrow \sim RL$:

1) $\exists x \in L$ when $|x| \geq n$

2) $\forall$ $\underbrace{uvw}_{\text{word}}$; such that $x = uvw$

3) $\exists\ i \geq 0$; uv^iw is [not] also L.

when word after pump up and pump down not in L so its $\sim RL$

4a) $|uv| \leq n$

4b) $|v| \geq 1$

$$L = \{a^n b^n ; n \leq 0\}$$

$L = \{\wedge, ab, aabb, aaabbb, \ldots\}$

$x = \underline{aabb}$ ✗ can't pick this. (restrict lenght 2)

$x = \underline{a^p b^p}$ $p \geq 0$ (means no automata of this L of any length)

Total P states.

$u: a^p$
$v: b^1$
$w: b^{p-1}$

$(p+1)$ $|uv| \leq n$ 4a)
not possible Should be less than p

$u: a^p$
$v: \wedge \longrightarrow$ 4b) $|v| \geq 1$ not possible.
$w: b^p$

$u: a^{p-1}$ $|uv| = (p)$ ✓
$v: a$ $|v| \geq 1$ ✓
$w: b^p$

$(a)^{p-1}(a)^i(b)^p$

$i = 2$ $(a^{p-1}(aa)b^p$ pump up ∈ L so L ∈ RL

$i = 0$ $(a)^{p-1}(b)^p \notin L$ pump down so L ∉ RL

$x = a^P b^P$

$u : a$

$v : a^{P-1}$] no upper bound on $|v| \geq 1$

$w : b^P$

$i=0 \quad ab^P \notin L$

$L \notin RL$.

② $\forall$ a.v. w

tells us to see all combinations.

but donot consider but logically x looping element x

$a^{P/2} b^{P/2}$ ↑

$\Rightarrow \frac{P}{2} + \frac{P}{2} = P$ valid

So, $x = a^P b^P$

$u = a^{P-s}$	$P \geq 0$
$v = a^s$	$P \geq s > 0$
$w = b^P$	

$x = a^{P-s} (a^s)^i b^P$

$i=0 \quad a^{P-s} b^P$ as $s \geq 1$

So, $s = 1$.

$a^{P-1} b^P$ so $\notin L$

so $L \notin RL$.

$L = \{0^n 1 0^n \; ; \; n \geq 0\}$

$L = \{1, 010, 00100, \cdots\}$

$x = 0^p 1 0^p$

$u = 0^1$

$v = 0^{p-1}$

$w = 10^p$

3/10/24

dependency exist

$L = \{a^n b^m \; ; n > m \;\&\; m \geq 0\}$

$n = m + c$

btw 2 or more variables, there is relationship or dependency then ~RL → need some kind of memory to proved relationship

$x = a^p b^q$ $\boxed{q \geq 1}$

$u = a$	a^{p-s}	
$v = a^{p-1}$	a^s	
$w = b^q$	b^q	

$q \geq 1$

$x = a^{p+q} b^p$

greater than this.

or $a^{p+1} b^p$ ✓ correct.

or

$x = a^{p+1} b^p$

considering one word.

p states → pumping length.

$u = a^p$
$v = a^1$ $\Big\}$ $= p+1 \not\leq 1$
$w = b^p$

$u = a^{p-1}$	a^{p-2}	p^{p-3}	
$v = a$	aa	aaa	
$w = ab^p$			

* v cannot be λ or u and w can be

introduces many divisions possible

$u = a^{p-s}$

$v = a^{s}$ $\quad p \geq s \geq 1$ ↳ lower bound

$w = a b^{p}$

$a^{p-s}(a^{s})^{i} a b^{p}$ $\quad i = 0$ pump down to disprove

$L = \{ a^{n} b^{2n} ; n \geq 0 \}$ ← imp

$a^{2n} b^{n}$

$x = a^{2p} b^{p}$

$u: a^{p-1} = a^{p-s}$

$v: a = a^{s}$ $\quad p \geq s \geq 1$

$w: a^{p} b^{p} = a^{p} b^{p}$

$a^{p-s} a^{1} b^{p}$

$a^{p-s+1} b^{p}$ if $\boxed{s = 1}$

$a^{p-1+1} b^{p}$ lower bound of s.

$a^{p} b^{p} \notin L$

so $L \not\subseteq RL$

$a^{p-s}(a^{s})^{i}(a^{p} b^{p})$ $\quad i = 0$ pump down

$a^{p-s+p} b^{p}$

$a^{2p-s} b^{p}$ let upper bound $s = p$

$a^{2p-p} b^{p}$

$a^{p} b^{p} \notin L \not\subseteq RL$

$a^{p-s} a^{2s} (a^{p} b^{p})$ $\quad i = 2$ pump up

$a^{p-s+2s} a^{p} b^{p} = a^{2p+s}$

$a^{2p+1} b^{p} \notin L \not\subseteq RL$ $\quad$ lower boud $s = 1$

* mostly use this. because we asume upper but lower is known.

$L = \{ n_a(x) = n_b(x) \}$

$a^{p} b^{p} b^{p} a^{p}$

Context Free Language:

two opening paranthesis ((so two closing)) paranthesis

$L = \{a^n b^n ; n \geq 0\} = \{\wedge, ab, aa\ bb, \ldots\}$

FA + stack ⇒ PDA

↓ pushdown automata. (machine)

powerful language than regular language.

we can write Grammer all structured work by context free grammer CFG

with finite automata and stack.

to check we use stack.

for(; ;) { // cout }

↓ infinite loop. like while(1) { }

→ in compiler it is e.g syntax

int a="helloworld"; syntax corred but semantic/context wrong

kind of recursive relation.

Grammer:

4 tuples $V, \Sigma, S, P \rightarrow$ Productions

V → variables, Σ → Terminal, S → Start variable equivalent to start state

Productions:

$A \rightarrow \alpha$ CFG

$A \in V$, $\alpha \in (V \cup \Sigma)^*$

$V = \{A\}$

$S = \{A\}$

$\Sigma = \{a, b\}$

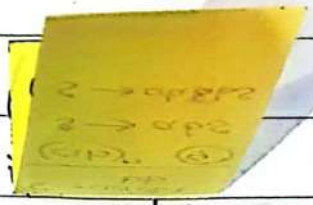
$A \rightarrow a^{\Sigma}$ $(V \cup \Sigma)^{\Sigma}$

A::a BNF

$A \rightarrow aA$

$(V \cup \Sigma)^2 = (V \cup \Sigma)(V \cup \Sigma)$

ΣV



```
void S(n) {
  if (n==0) return ;   ① base case
  ② recursive relation
  S(n-1)
  cout << "a";
  cout << "b";
}
```

$\rightarrow a^n b^n$

function call.

$V = \{S\}$

$\Sigma = \{a, b\}$

$S_V = S$

Production:

$S \rightarrow aSb$

1 rule in grammer based on ②

$S \rightarrow \wedge$ ①

```
return ;
count = n
while(count)
  cout << a
  count--
count = n
while (count)
  cout << b
  count--
```

not recursion.

$S(n-1)$

$a^n b^n$ $n > 1$ ②

$S \rightarrow aSb$

$S \rightarrow aabb$

① $a^n b^n$ $n \geq 0$

base case change if (n==1) cout "ab" $S \rightarrow ab$

ntax correct
emantic / contextwrong

while(1)
{ }
kind of recursive relation.

Σ , S , P → Productions

Terminal Start variable
equivalent to start state

V = {A}
S = {A}
Σ = {a, b}

A → a (V ∪ Σ)* Σ

A ::= a BNF

→ $a^n b^n$

function call.

V = {S}
Σ = {a, b}

= "a"; S_V = S

Production:

S → aSb

1 rule in grammer based on (2)

S → Λ (1)

$a^{2n}\ b^n$ (3)

S → aaSb

S → aaaa bb

$(ab)^n$ (4)

S → abS

S → ababS

not recursion.

1 (2)

Theory of Automata :- CFG

$L = \{a^i b^j ; i < j ; j \geq 0\}$

Grammer :- 4 tuples $\{V, \Sigma, S, P\}$

$L = \{b, bb, bbb, abb, \}$

so not Possible

Start V = S

①	$S \to b \cdot B$	$L = \{b^+\}$
②	$B \to Bb$	$bbb \in L(G)$?
③	$B \to \lambda$	

no difference

right side → terminals or variables.

$V = \{S, B\}$

$\Sigma = \{b, \lambda\}$

$S_V = S$

DFA → CFG
R.E → CFG

b^*

$B \to \lambda / bB$
$B \to \lambda$
$B \to bB$

$bb^* = b^+$

$B \to b$ — ①
$B \to Bb$ — ②

$B \to b / Bb$
① ②

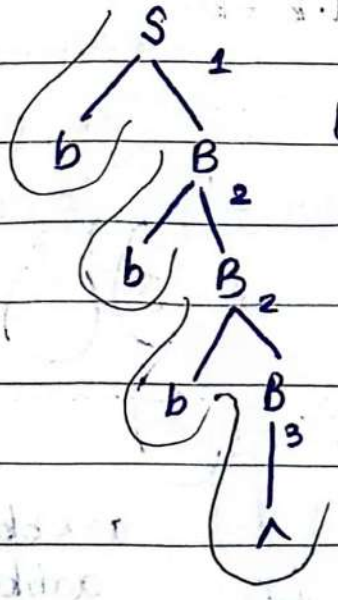
logical OR operator.

production : $A \to \alpha$
$A \in V$
$\alpha = (V \cup \Sigma)^*$

this cause λ in terminal Σ

String Processing

1) Parse Tree



leaf nodes must be a terminal or null λ. (Σ)

Top to bottom
Left to Right

$bbb\lambda = bbb$

2) Substitution:-

$$L(G) = \{x \in \Sigma^* \mid S \overset{*}{\Rightarrow} x\}$$

* → rule no
x → word
G → Grammer

$$S \overset{1}{\Rightarrow} bB \overset{2}{\Rightarrow} bbB \overset{2}{\Rightarrow} bbbB \overset{3}{\Rightarrow} bbb\Lambda$$

we cannot stop derivation process because it has a variable.

$a^n b^n$ ← $a^i b^j$; $i \geq 0$, $j \geq 1$

$B \to b^+$

RE = $bb^* + a$

① $S \to bB \mid a$ ④

② $B \to bB$

③ $B \to \Lambda$

$x = aaab$

$L = \{a^i b^j ; i < j ; j \geq 0\}$

④ $A \to aAb \mid \Lambda$ ⑤

② ③ $B \to b \mid bB$ b^+

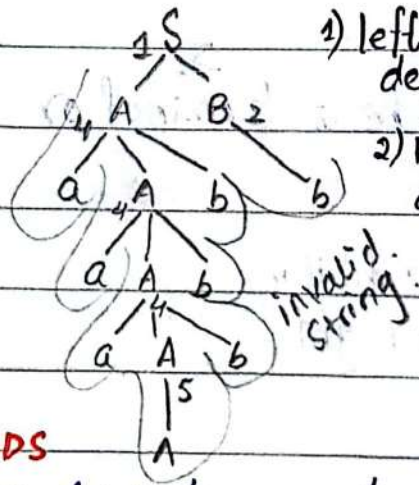
$S \to AB$ ①

abb
aabbb

minimum word of each
$\Lambda . b = b$

invalid str

$x = aaab$



1) left most derivation tree

2) Right most derivation tree

invalid string.

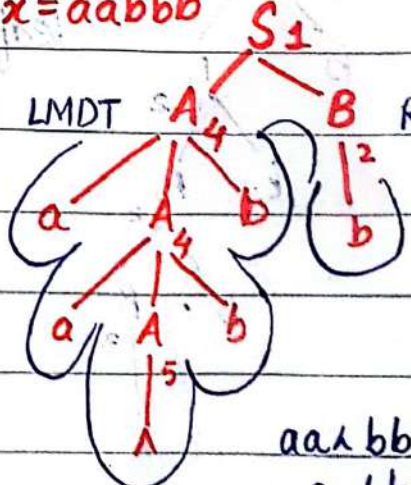
LMDS

$S \overset{1}{\Rightarrow} AB \overset{4}{\Rightarrow} aAbB \overset{4}{\Rightarrow} aaAbB \overset{5}{\Rightarrow} aa\Lambda bB$

$\overset{2}{\Rightarrow} aa\Lambda bbb$

$aabbb$

$x = aabbb$



complete LMDT

$aa\Lambda bbb$
$aabbb$
valid string

RMS: (Right most substitution) start from suffix.

$S \Rightarrow^{1} AB \Rightarrow^{2} Ab \Rightarrow^{4} aAbb \Rightarrow^{4} aaAbbb \Rightarrow^{5} aa\wedge bbb$

$aabbb$

$L = \{ a^i b^j \; ; \; i > j \; ; \; i <= 0 \}$

$a^k a^i b^i \rightarrow a^n b^n$ (B)

a^+ (A)

$L = \{ a, aa, aaa, aabb, \ldots \}$

$a^+ \quad a^n b^n$

① $S \rightarrow AB$

④ $B \rightarrow aBb \mid \wedge$ ⑤

② $A \rightarrow a \mid aA$ ③

$L = \{ a^i b^j \; ; \; i \geq j, \; i <= 0 \}$

$a^k a^i b^j$ | $a^i b^i$

$L = \{ \wedge, a, ab, aa, \ldots \}$

① $S \rightarrow AB \mid B$

② $A \rightarrow a \mid aA$ ③

④ $B \rightarrow aAb \mid \wedge$ ⑤

$L = \{ a^i b^j \; ; \; i \neq j \; ; \; i \geq 0 \}$

$a^i b^i b^k \mid a^i a^k b^i$

① $S \rightarrow AB \mid CD$

$A \rightarrow a \mid aA$

$B \rightarrow aBb \mid \wedge$

$C \rightarrow b \mid bC$

$D \rightarrow aDb \mid \wedge$

10/10/24

$L = \{a^i b^j ; i <= j \text{ or } i = j ; i >= 0\}$

$i \geq 0 \quad k \geq 1 \quad a^i b^i b^k \mid a^i b^j$

$L = \{\wedge, b, ab, bb, abb, bbb, aabb, \ldots\}$

① $S \rightarrow AB \mid A$

$A \rightarrow aAb \mid \wedge$

$B \rightarrow bB \mid b$

(base cases)

$S \rightarrow \wedge \mid b \mid Sb \mid aSb$

Sb → all related combinations of b's

→ a and b can be equal

* recursion types
→ indirect
→ nested
→ direct

mathematic decomposition OR
1) See language
2) Extract pattern
3) Write CFG

OR

② $S \rightarrow \underset{①}{\wedge} \mid \underset{②}{Sb} \mid \underset{③}{aSb}$ → when on same level position of terminal matters alot

if $S \rightarrow \underset{①}{\wedge} \mid \underset{②}{bS} \mid \underset{③}{aSb}$

$\rightarrow \wedge bab$ (incorrect)

mathematical decomposition

$i <= j$

$i < j$ (A) | $i = j$ (B)

$S \rightarrow A \mid B$ $a^n b^n$

$B \rightarrow aBb \mid \wedge$

$S \rightarrow A \mid B$

$B \rightarrow aBb \mid \wedge \rightarrow a^n b^n$

$A \rightarrow BD \rightarrow i < j$

$D \rightarrow bD \mid b \rightarrow b^+$

$A \rightarrow BA \mid Ab \mid b$ b^+

causes ↓ abab

A
↓
BA
↓ ↓
ab BA or BA
↓
ab

$L = \{ a^i b^i c^j d^j \; ; \; i>=2 \; ; \; j>=3 \}$ — $i=2$ $i<2$ $j=3$ $j<3$

aa bb ccc ddd — Base cases.

$L = \{ \wedge, abcd$

$S \to AC \mid BD$

$A \to aaAbb \mid aabb$ — $i=2$

$C \to cccCddd \mid cccddd$ — $j=3$

$B \to aBb \mid \wedge$ — $i<2$

$D \to cDd \mid \wedge$ — $j<2$

$S \to AC \mid \wedge$

$A \to aaAbb \mid aabb$

$C \to cccCddd \mid cccddd$

$L = \{ \underbrace{a^i b^i}_{A} \; \underbrace{c^j d^j}_{C} \; ; \; i>=2 \; ; \; j>=3 \}$ — Base cases

$L = \{ aabbccddd, \quad i>2 \mid i=2 \quad j>3 \mid j=3.$

$S \to AC$

$A \to aAb \mid aabb$

$C \to cCd \mid ccc ddd$

$L = \{ \underbrace{a^{2i} b^i}_{A} \; \underbrace{c^j d^{3j}}_{B} \; ; \; i<=2 \; ; \; j<=3 \}$

$S \to AB \mid \wedge$

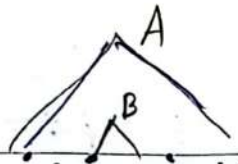
$A \to aaaabb \mid aaAb$

$B \to cccdddddddddd \mid cBddd.$

$S \to AB$

$A \to \wedge \mid aabb \mid aaaabb$

$B \to \wedge \mid cddd \mid ccddddddd.$



$L = \{ a^i b^j c^j d^i \quad i>=2 \quad j>=3 \}$

A C

$i>2 \mid i=2 \quad j>3 \mid j=3$

$S \rightarrow ABC$

$A \rightarrow aA \mid aa$

$B \rightarrow bBC \mid bbbccc$

$C \rightarrow dD \mid dd$

$A \rightarrow aA \mid B\,dD \mid aabbbccdd$

$B \rightarrow bBc \mid bbbccc$

Answer

$S \rightarrow aSd \mid aaBdd$ nested recursion.

$B \rightarrow bBc \mid bbbccc$

OR

$A \rightarrow aAd \mid aaBdd$

$B \rightarrow bBc \mid bbbccc$

15/10/24

$L = \{ n_a(x) = n_b(x) \}$ → order do not matter

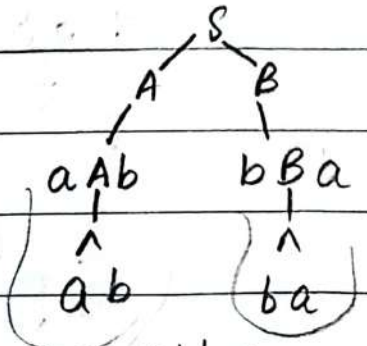
$a^n b^n$ $b^n a^n$ eg $a^i b^n a^{n-i}$

$S \rightarrow AB$ $(S \rightarrow A|B)^*$

$A \rightarrow aAb|\wedge$

$B \rightarrow bBa|\wedge$

abba



= abba (accept)

incorrect

$a^n b^m$ / $b^m a^n$

$A \rightarrow aAb|\wedge$

$b^* \leftarrow B \rightarrow bB|\wedge$

$C \rightarrow bCa|\wedge$

$a^* \leftarrow C \rightarrow aC|\wedge$

$S \rightarrow A$

but abbbaa

$a^n b^n$

$a^1 b^3 a^{3-1}$

$a^i b^n a^{n-i}$

$S \rightarrow \wedge | aSb | bSa$

$a^n b^n$ $b^n a^n$ ordered

$L = \{ n_a(x) = n_b(x) \}$ no pattern exists.

$S \rightarrow \overset{1}{aSb} | \overset{2}{bSa} | \overset{4}{\wedge} | \overset{3}{SS}$

rejected

abbbaa

$S \overset{1}{\Rightarrow} aSb \overset{1}{\Rightarrow} aaSbb$

start, end

$\overset{2}{\Rightarrow}$ abSab

ababbaab

abaSaab

(issue)

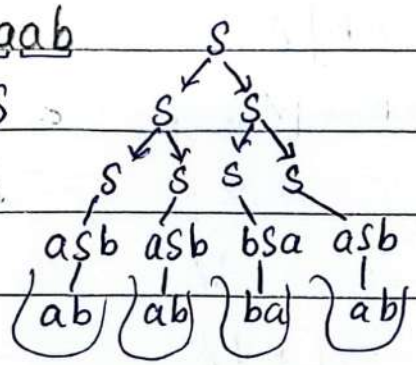
$S \overset{3}{\Rightarrow} SS \overset{1}{\Rightarrow} aSbS \overset{2}{\Rightarrow} aSbbSa \overset{4}{\Rightarrow} a\wedge bbSa \overset{4}{\Rightarrow} \boxed{abba}$

accept

$S \to \overset{1}{a}Sb \mid \overset{2}{b}Sa \mid \overset{4}{\wedge} \mid \overset{3}{SS}$

ababbaab

S S S S



= ababbaab (accept)

$S \to ab$

a b ab (equality)

a a aa (need 2 b's)

$S \to \overset{1}{\wedge} \mid \overset{2}{aB} \mid bA \mid$ → need (b)

$B \to bS^{3}$ $A \to aS$

↓ come back to level

2,3,1,2,3,1 $ab\wedge ab\wedge$

$ab\wedge ab\wedge = \boxed{abab}$

abba accept

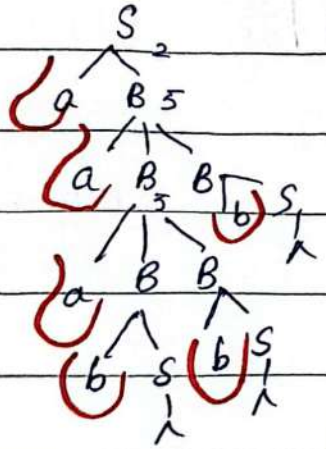
$S \to \overset{1}{\wedge} \mid \overset{2}{aB} \mid \overset{3}{bA} \mid$

$B \to \overset{4}{b}S$

$A \to \overset{5}{a}S$

← abbbaa

cannot be produced.



S0,

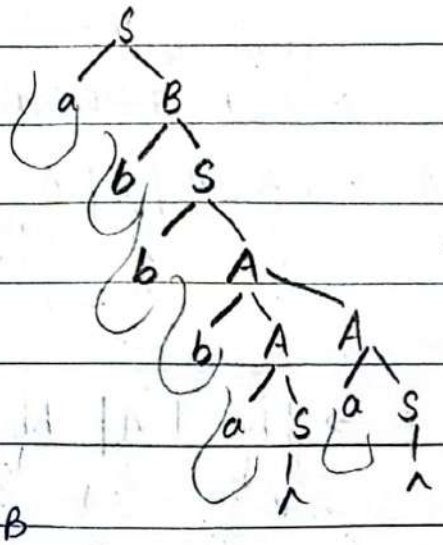
$S \to \overset{1}{\wedge} \mid \overset{2}{aB} \mid \overset{3}{bA}$

$B \to \overset{4}{b}S \mid \overset{5}{aBB}$

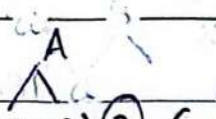
$A \to \overset{6}{a}S \mid \overset{7}{bAA}$

$\boxed{aaabbb}$

abbbaa



= abbbaa



$L = \{ (a^n b^n)^p (c^m d^m)^p \text{ same} , n, m, p \geq 0 \}$

↳ sequence exists

$S \rightarrow \lambda \mid ASB$

$A \rightarrow aAb \mid \cancel{AA} \mid \lambda$

$B \rightarrow cCd \mid \cancel{BB} \mid \lambda$

$S \rightarrow \lambda \mid AS \mid SB \mid$ — can come alone so wrong.

Ans: $S \rightarrow ASB \mid \lambda$

$A \rightarrow aAb \mid \lambda$

$B \rightarrow CBd \mid \lambda$

abab cd (λ) prob due to base.

if m, n, p > 0 then might accept.

↳ cannot make sure by single stack, need more than one stack.

abaabb ccddddcd

cannot check this.

out of scope

Simple and cute ;)

variable exactly 2

at start only

CNF:

(X) 3 variable

$V \to VV \mid \Sigma$ (only one terminal)

$S_V \to \Lambda$

$S \to {}^{1}ASB$

CFG

$A \to {}^{2}aA\,SA \mid {}^{3}a \mid {}^{4}\Lambda$

Σ (1) V (3) (X)

terminal (✓)

null at intermediate (X)

$B \to {}^{5}SbS \mid {}^{6}A \mid {}^{7}bb$

V (2) V (1) (in CNF)

V (1) x

Σ (2) x

chomsky normal form

overall not in CNF

CNF used for proving pumping lamma (grammer should be in CNF)

binary tree.

Step#1:

Create a New Start variable

$S_V \to S$

won't be on right side of grammer or reproduced again.

Step#2:

Remove Λ productions.

find A substitute Λ cuz

$A \to \Lambda$

Rule 1: $ASB = \Lambda SB = SB$ new Rule

so, $S \to ASB \mid SB$

$A \to aASA \mid a \mid aSA \mid aAS \mid aS$

multiple locations (check every combination of Λ)

$a\Lambda SA = aSA$

$aASA = aAS$

$a\Lambda S\Lambda = aS$

$B \to SbS \mid A \mid bb \mid \Lambda$

single A so Λ

So, again do for B

CamScanner

Remove ∧ from B

$S_0 \to S$ (1)

$S \to ASB$ (2) $| SB$ (3) $| AS$ (4) $| S$ (5)

$A \to @ASA$ (6) $| a$ (7) $| @S$ (8) $| @SA$ (9) $| @AS$ (10)

$B \to SbS$ (11) $| A$ (12) $| bb$ (13)

Step 3: Remove unit production $V \to V$

just 1 variable only. $V \to P$, $V \in V$, $P \in V$

→ Remove Self calls.

→ if we remove A (12) from B

all rules of A

$B \to SbS \mid aASA \mid a \mid aS \mid aSA \mid aAS \mid bb$

$S_0 \to ASB \mid SB \mid AS \mid \not{S}$ (self calls removed to avoid loop.)

Step 4: **Create new Variable** to bring in CNF format.

$P \to a$ ($V \to \Sigma$) $\quad Q \to b \quad R \to AS$ cant make SA as R.

$S \to ASB \mid SB \mid AS$

$A \to PASA \mid a \mid PS \mid PSA \mid PAS$

$B \to SQS \mid QQ \mid PASA \mid a \mid PS \mid PSA \mid PAS$

$S \to RB \mid SB \mid AS$

$A \to PRA^{u} \mid a \mid PS \mid P\overset{v}{S}A \mid PR$

$B \to S\overset{w}{Q}S \mid QQ \mid PR\overset{u}{A} \mid a \mid PS \mid P\overset{v}{S}A \mid PR$

$U \to RA \quad V \to PS \quad W \to SQ$

$S \to RB \mid SB \mid AS$

$A \to PU \mid a \mid PS \mid VA \mid PR$

$B \to WS \mid QQ \mid PU \mid a \mid PS \mid VA \mid PR$

CamScanner

17/10/24 CFG → CNF

$S \rightarrow \{S\} \mid \wedge$

$S_0 \rightarrow S$ ① make S_0

$S \rightarrow \{S\} \mid \wedge$ ② remove $\wedge$

$S \rightarrow \{S\} \mid \{\}$ ③ remove unit productions

$S_0 \rightarrow S \mid \wedge$

$S_0 \rightarrow \wedge \mid \{S\} \mid \underbrace{\{\}}_{\Sigma\Sigma}$ ②

$S \rightarrow \{S\} \mid \underbrace{\{\}}_{\Sigma\Sigma}$

$P \rightarrow \{^{6}$

$Q \rightarrow \}^{7}$ ④ make new variable.

$\{S\}$ ΣVΣ

↳ not a unit call because it has terminals in it.

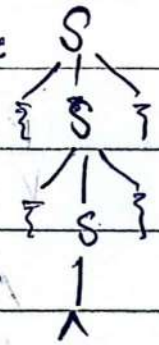
$S_0 \rightarrow \wedge \mid PSQ \mid PQ$

$S \rightarrow PSQ \mid PQ$

$L = \{\{^n \}^n ; n \geq 0\}$

Input = {{}}

Topdown:

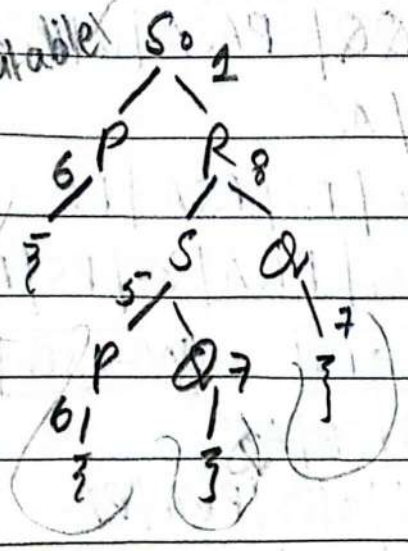


$R^{8} \rightarrow SQ$

$S_0 \rightarrow \wedge^{3} \mid PR^{4} \mid PQ^{2}$

$S \rightarrow PR^{4} \mid PQ^{5}$

not suitable



CYK-Parser

CYK-Parser

must need CNF not CFG

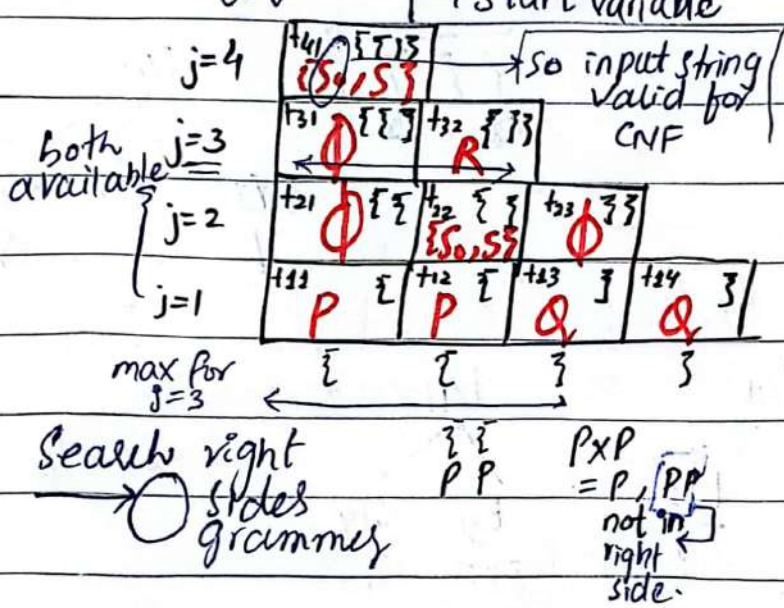
input → start variable

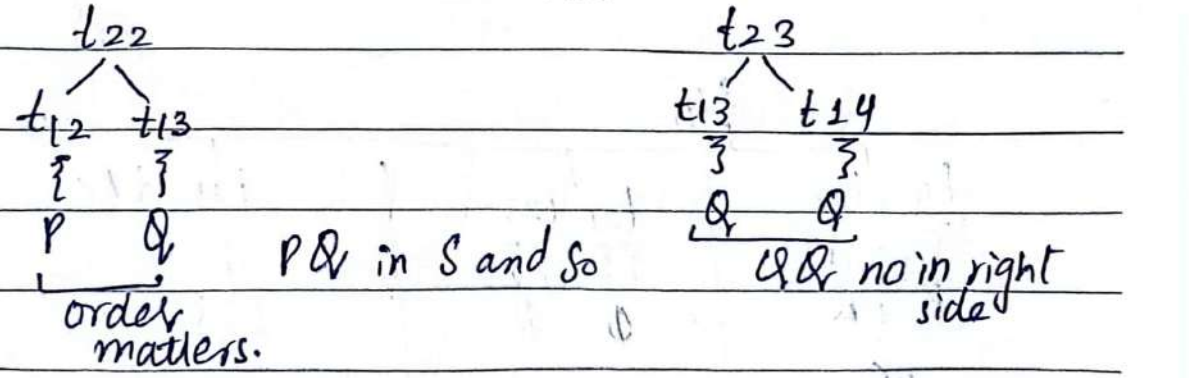
j=4, j=3, j=2 (bottom up), j=1

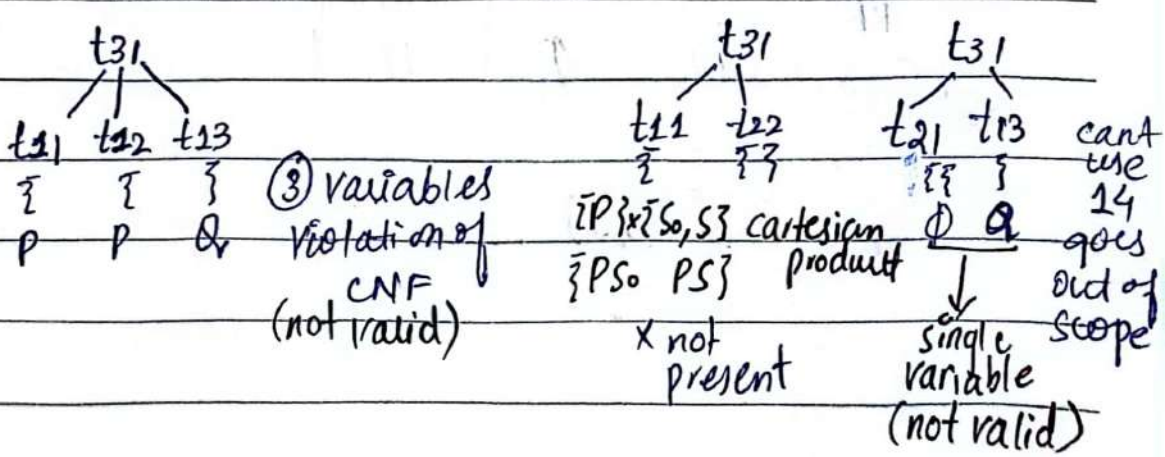
substrings (y)

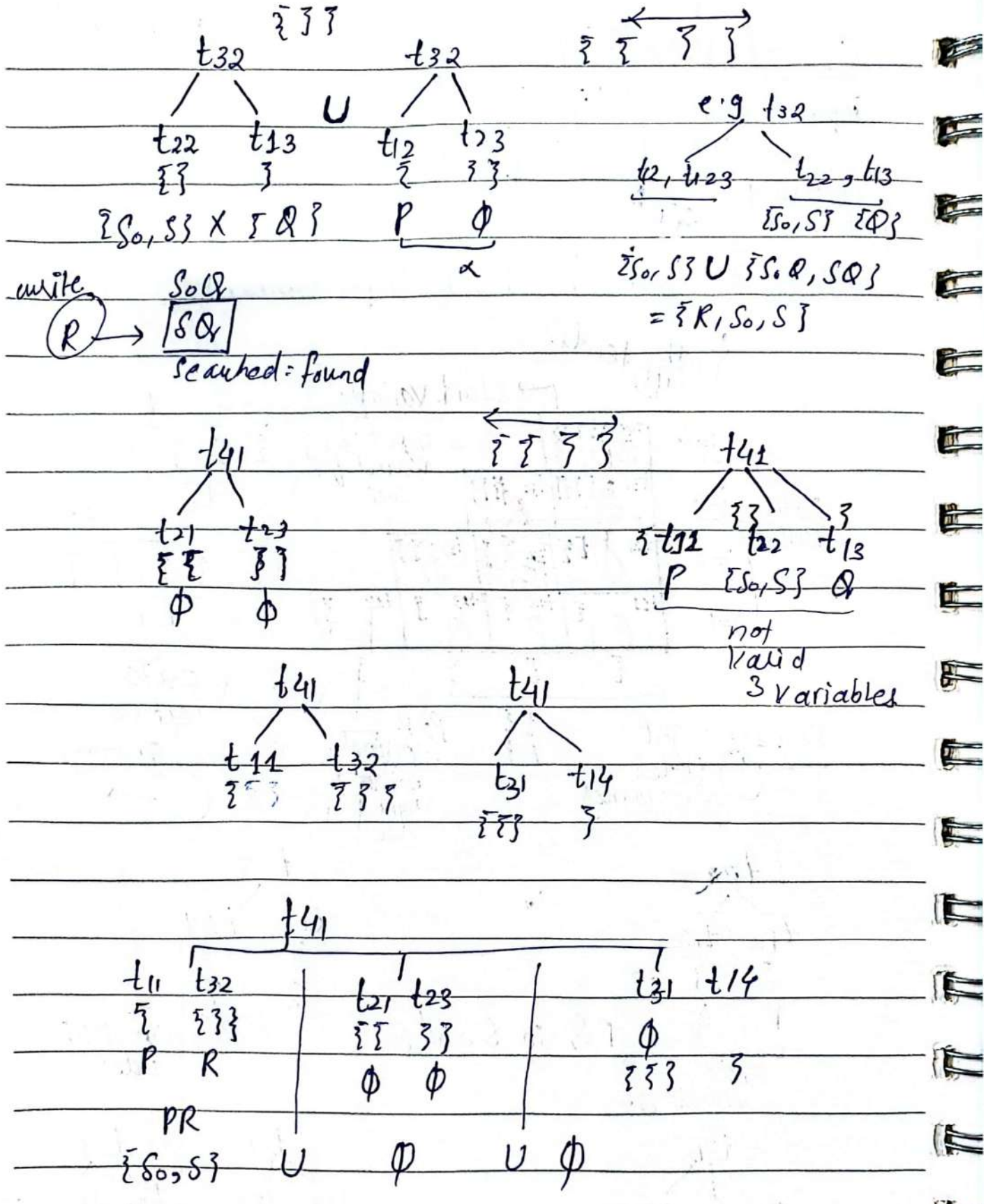
with diff levels.

(input x axis)








t32
t22
t13
U
t12
t23
{So, S} X {Q}
P
ϕ
α
e.g t32
t2, t23
t22, t13
{So, S} {ϕ}
{So, S} U {So, Q, SQ}
= {R, So, S}
write
SoQ
R
SQ
searched: found
t41
t21
t23
ϕ
ϕ
t41
t11
t22
t13
P
{So, S}
Q
not valid
3 variables
t41
t11
t32
t41
t31
t14
t41
t11
t32
P
R
PR
{So, S}
U
ϕ
t21
t23
ϕ
ϕ
U
ϕ
t31
t14
ϕ
= {So, S}

Push Down Automata :-

PDA

FA — Tuple = 5 — $\delta, Q, \Sigma, f_0, q_0$

Stack — Tuple = 2 — Γ, Z_0

Γ → stack element

Z_0 → top element of stack. (marker ($))

$\Gamma = \{\$, A\}$ — A: any symbol in stack.

aaa bbb

push pop

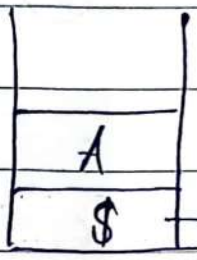
A
A
A
$

equal only if start with $ sign and end up with $ sign

Format:

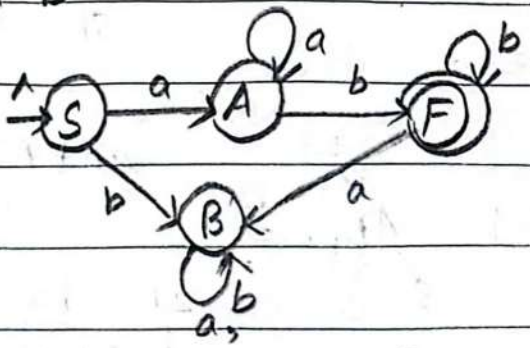
$\Sigma, \Gamma/\Gamma$

↓ pop ↓ push



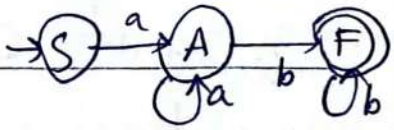
if only $ empty stack.

a^+b^+

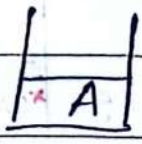


DFA

a^+b^+ NFA :-



a, $/A (replace op)



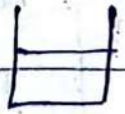
a, $/$ (no op)

a read (pop)

$ extract

$ push.

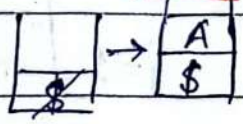
a, $/Λ (pop)



(push)

a, $/A$

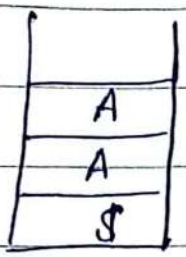
R-L push.

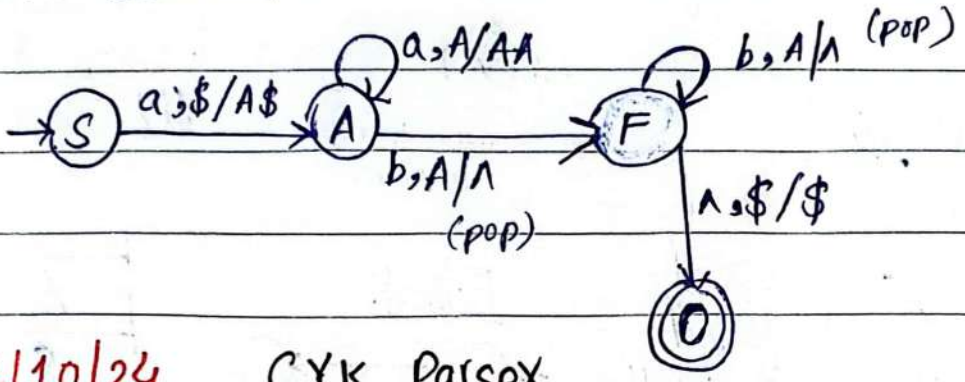


seek (only read) Cipser

extract (pop) Martin

multipush a , $/ A A $





22/10/24 CYK Parsex

CNF
- no unit product
- no Λ in intermidiate
- no S or vars on right side

$S \to AB^{1} \mid BC^{2}$

$A \to BA^{3} \mid a^{4}$

$B \to CC^{5} \mid b^{6}$

$C \to AB^{7} \mid a^{8}$

if $S \to AB \mid BC \mid SB$ (SB: no in CNF)

so, $S_0 \to S$

steps to make SNF

baaba

CB ≠ BC

23 → AB (S, C)

start variable so valid string belong to it

j=5	51 baaba {S,A,C}				
j=4	41 baab Φ	42 aaba {S,A,C}			
j=3	31 baa Φ	32 aab {B}	33 aba {B}		
j=2	21 ba {S,A}	22 aa {B}	23 ab {S,C}	24 ba {S,A}	
j=1	11 b {B}	12 a {A,C}	13 a {A,C}	14 b {B}	15 a {A,C}

5 cells 11, 12, 13, 24, 22, 3

b a a b a

B → b A → a C → a

21 ba → 11 12

{B} × {A, C}

{BA, BC}

{A ∪ S}

= {S, A}

aa → 22 → 12 (A, C) | 13 (A, C)

AA, AC, CA, CC → B

baa

→ if we combine 11, 12, 13

3 variable combination not a CNF

{B} × {A,C} × {A,C}

{B,A,C}

31

(1) / (2) 11, 22 — b | aa

BB × ϕ

(2) (1) 21, 13 — ba | a

{S,A} {A,C}

SA AA × ϕ

SC AC

aab 32

22, 14 — aa | b

{B} {B}

BB ϕ

12, 23 — a | ab

{A,C} {S,C}

AS CC

AC CS

{B}

aba

23, 15 — ab | a

{S,C} {A,C}

SA CC

SC CA

{B}

13, 24 — a | ba

{A,C} {S,A}

AS CS

AA CA

× ϕ

Q1:

baab

(3) / (1) 31 | 14

ϕ {B}

= B no in right side

ϕ ∪ = ϕ

baab

(1) / (3) 11 | 32

{B} {B}

BB

∪ = ϕ

baab

(2) / (2) 21 | 23

{S,A} {S,C}

SS AS

SC AC

∪ = ϕ

1 ϕ so no need to check another

= {ϕ}

Q2: aaba

32 — aab | 15 — a

{B} {A,C}

BA BC

{S,A} ∪

aaba

12 — a | 3 — aba

{A,C} {B}

AB CB

= {S,C} ∪

aaba

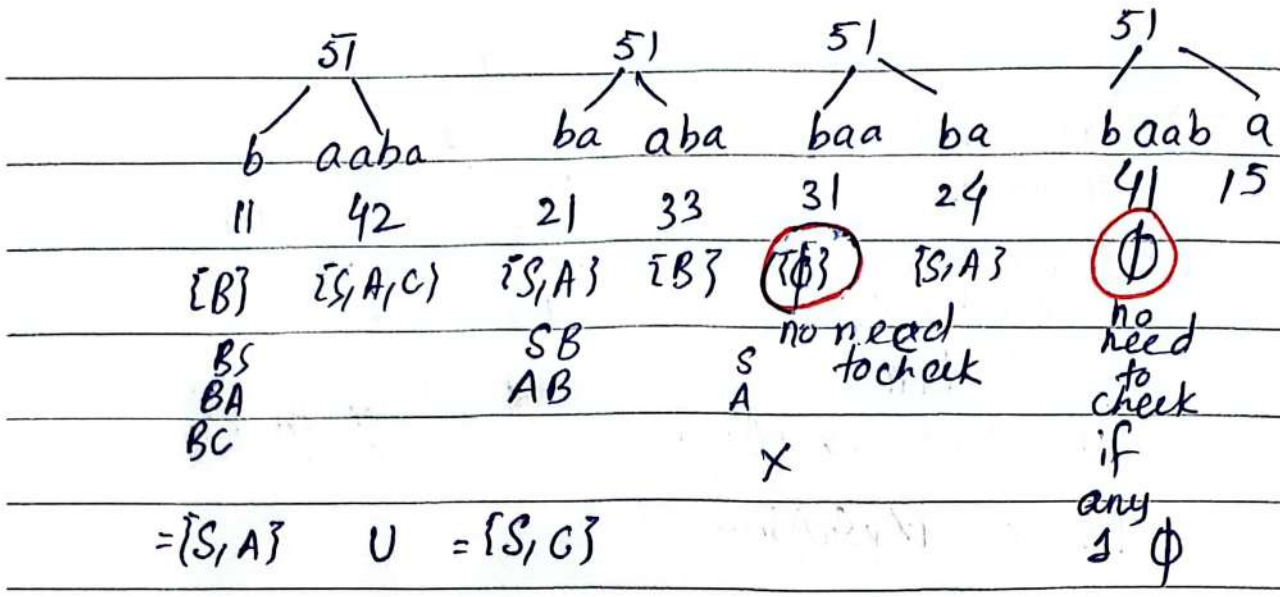
22 — aa | 24 — ba

{B} {S,A}

BS BA

= {A}

= {S, A, C}



$= \{S, A\} \cup = \{S, C\}$

$\{S, A, C\}$

$n = 5$ table 5×5 2D dynamic array

→ space $O(n^2)$

→ time complexity $O(n^3)$

24/10/24 PDA

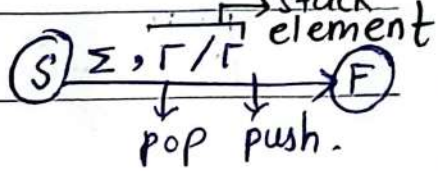
$L = \{ a^n b^m \; ; \; n > m \; ; \; m \geq 0 \}$

$L = \{ a, aa, aaa, aab, aaaa, \dots \}$ $\quad \Sigma = \{a, b\}$

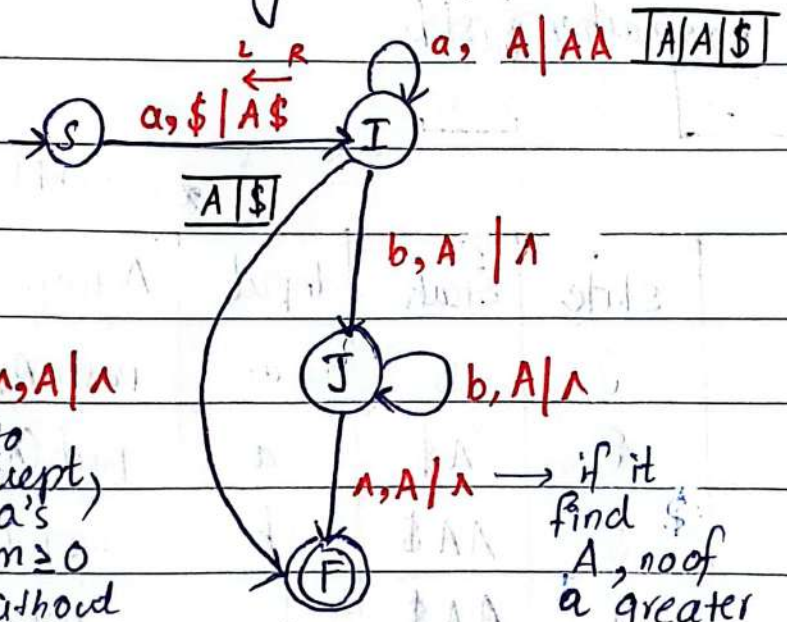
$\Gamma = \{ \$, A \}$

→ 4 transitions at each state skipped transition maps to dead state automatically

→ right most goes to stacks first



→ Stack → only see top element



(to accept, a's $m \geq 0$ without this arrow $m > 0$)

→ if it find A, no of a greater if $ then a and b equal.

Transition Formula

$Q \times \boxed{\Sigma \times \Gamma} \rightarrow Q \times \Gamma$

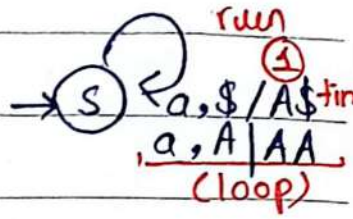
→ Stack plus 1 at every state or transition.

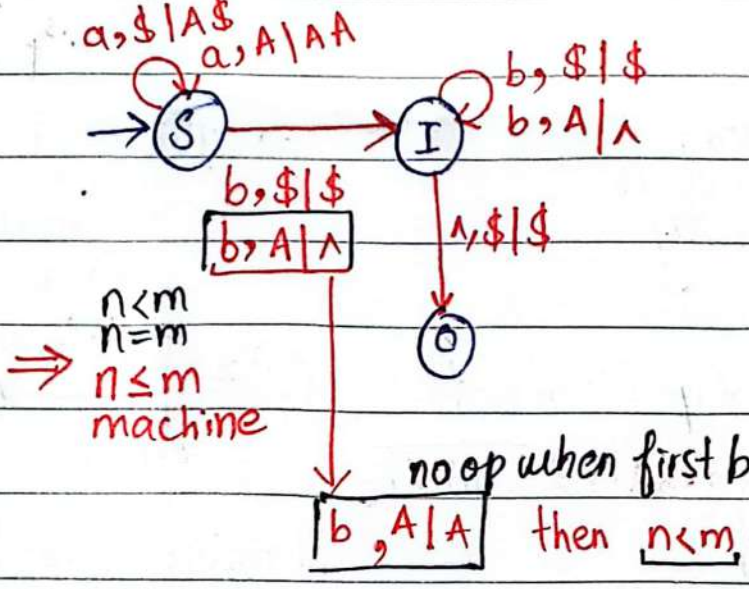
aab ∧ at end then acceptance.

$L = \{ a^n b^m ; n<m ; n \geq 0 \}$

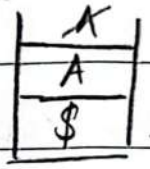
$L = \{ b, bb, bbb, abb, \dots \}$
$abbb, aabbb, \dots$

$n<m$
1 2





$x = aabb$



aabb
so, it won't accept

aabbbbb

$x = aabbb$

	state	Stack	Input	Action
	S	$	a	push(A), S
A a b b	S	A$	a	push(A), S
A a b	S	AA$	b	no-pop, I
$ b b λ	I	AA$	b	pop(), I
	I	A$	b	pop(), I
	I	$	λ	Accept
				no op

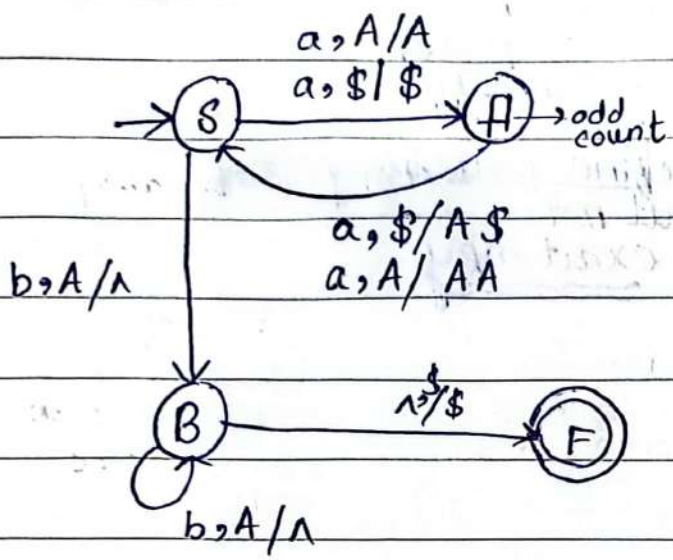
$a^n b^{2n}$ *

$L = \{ a^{2n}, b^n ; n \geq 1 \}$

$L = \{ aab, aaaabb, aaaaaabbbb, \ldots \}$

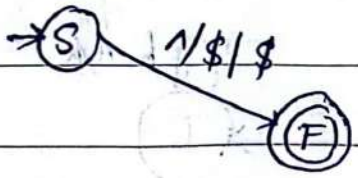
method#1:

if two aa push A in stack

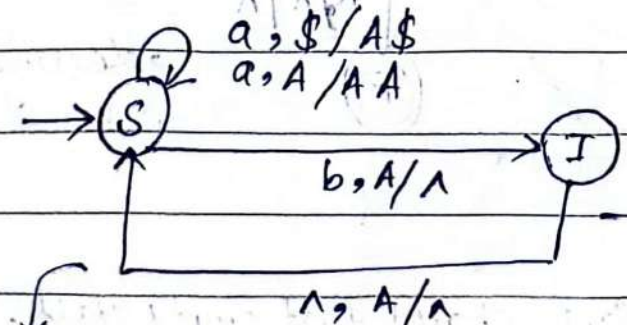


*no multiple pop in Stack.

if $n \geq 0$ ∧ accept

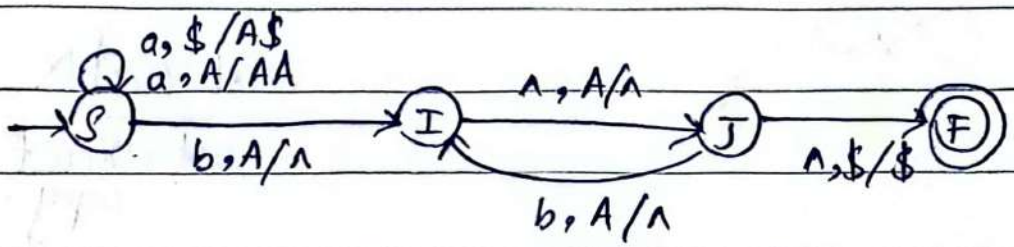


method#2:



for multiple pop.

pattern violate = ab∧ab



29/10/24 PDA

$L = \{ (a^n b^m)^i ; n > m ; i, m > 0 \}$

→ Pattern can repeat

actual value of n and m in power i can differ.

it can define pattern but not exact copy

m min = 1

n = 2

i = 2

aaaab abb

n < m ∝

$a^n b^m$ $a^n b^m$

i = 2

aaaab aaabb

n = 4 m = 1 | n = 3 m = 2

i = 1

m = 1 n = 2

$(aab)^1 = aab$

m = 2 n = 3

$(aaabb)^1 = aaabb$

i = 2

m = 1 n = 2

m = 2 n = 3

(aab aaabb)

No Op

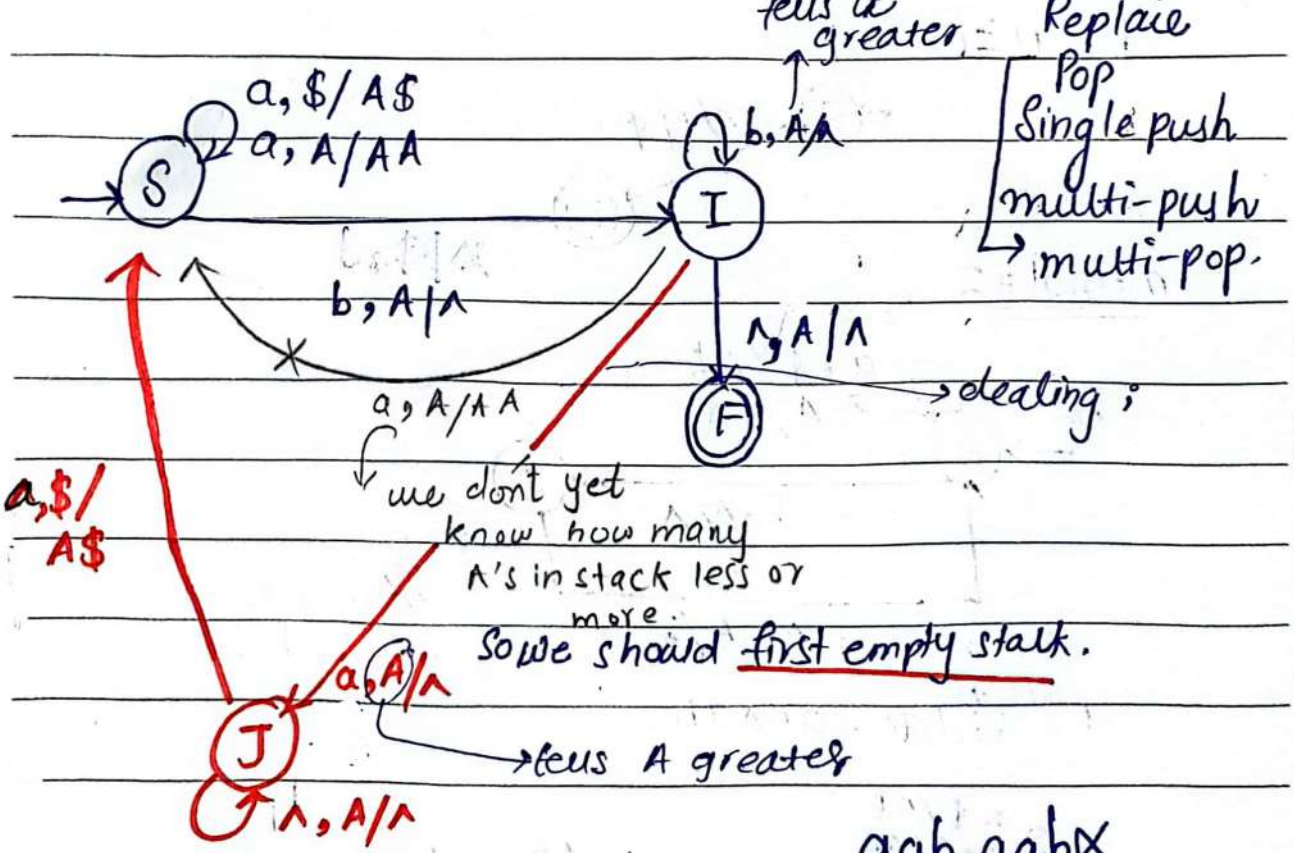
Replace

Pop

Single push

multi-push

multi-pop.

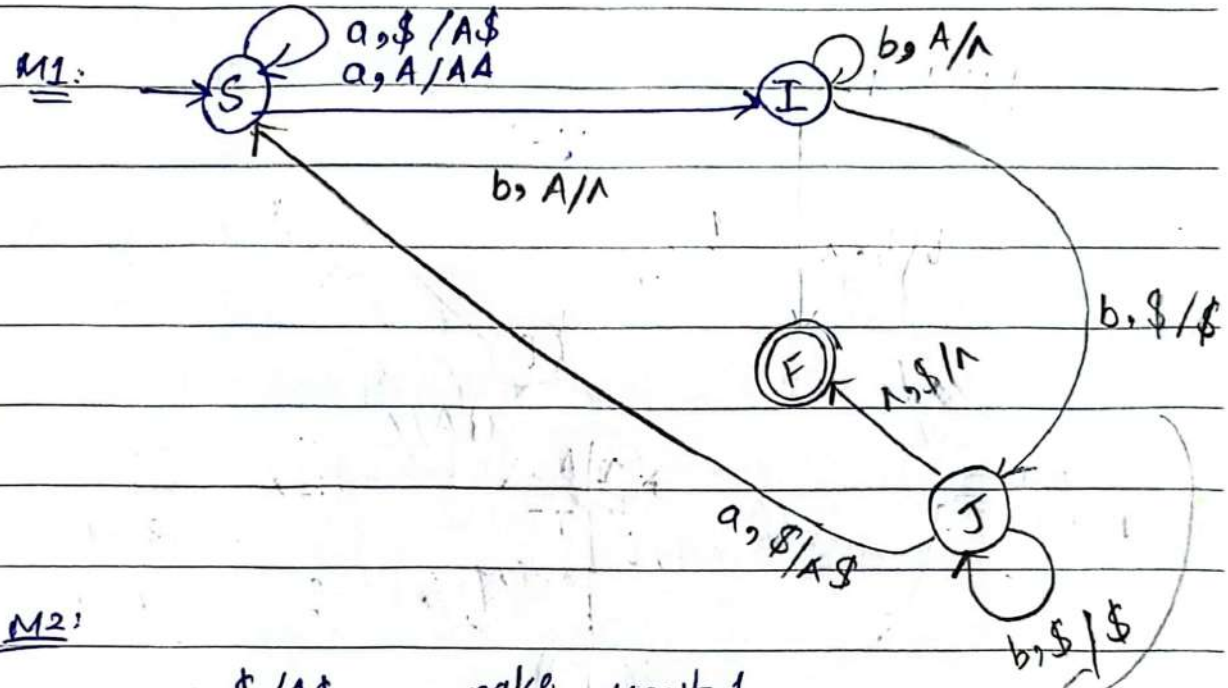


aab aabx

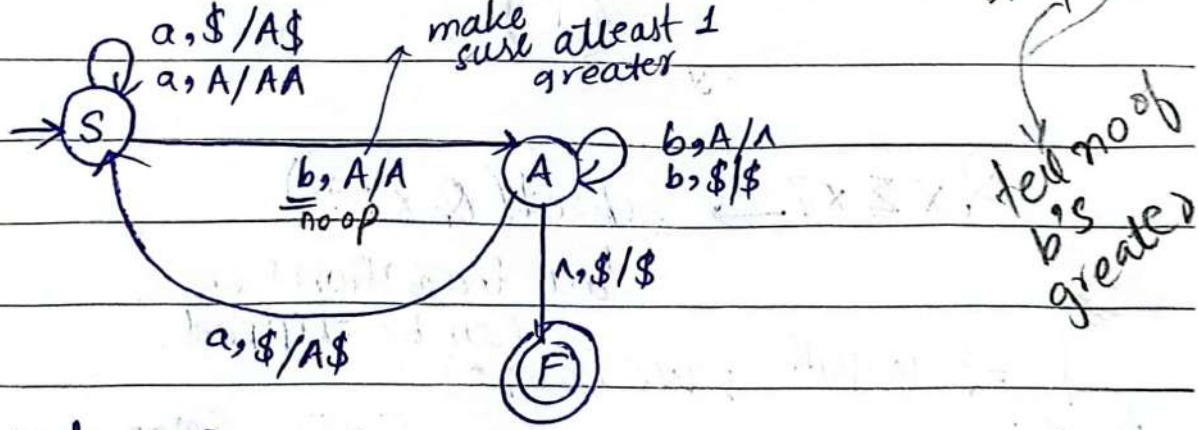
not exact copy.

$L = \{ (a^n b^m)^i \; ; \; n < m \; ; \; n, i > 0 \}$ aabbb

M1:



M2:



palindrome :

$L = \{ W C W^R \; ; \; w \in \Sigma^* \}$ $\Sigma = \{a, b\} \cup \{c\}$

reverse

marker

$\{a, b\}^*$

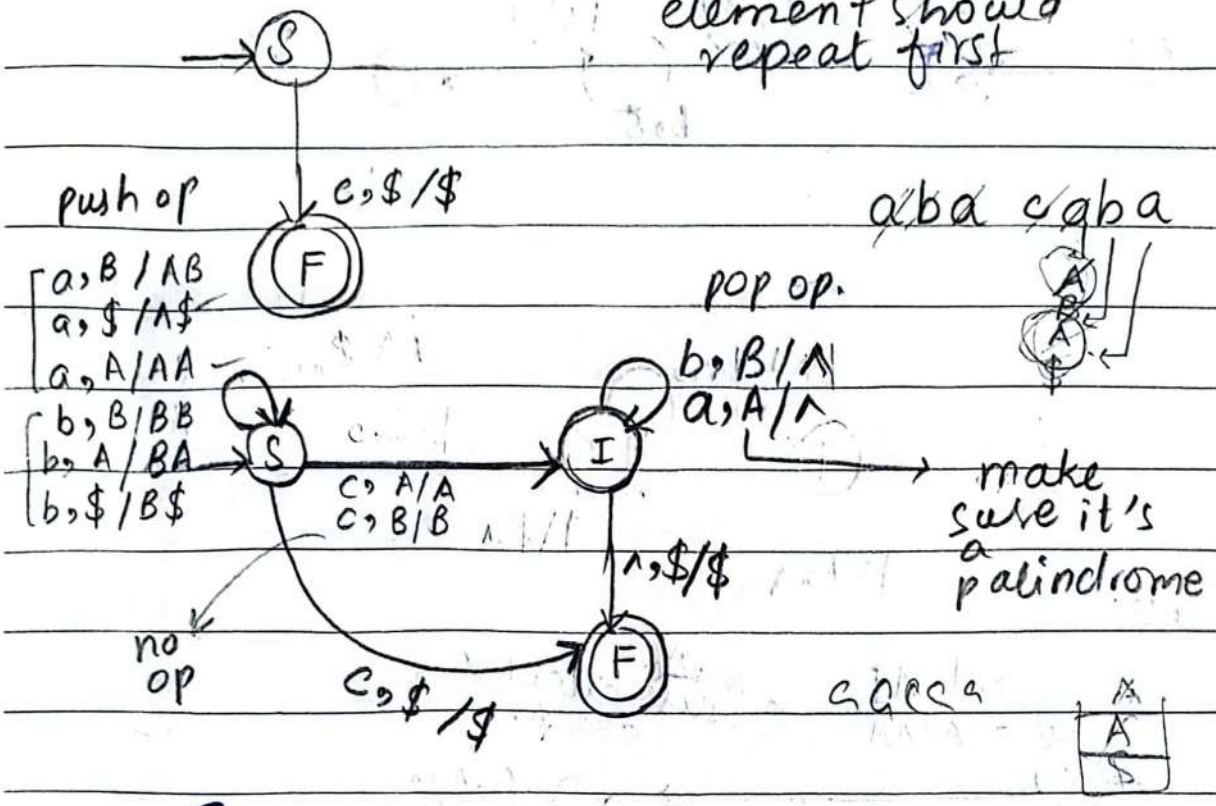
in middle of word as marker

language of palindrome

$L = \{ c, aca, bcb, abcba, bacab, aacaa, bbcbb, aaacaaa, aabcbaa, abacaba, \ldots \}$

DPDA Deterministic PDA

→ stack top element should repeat first



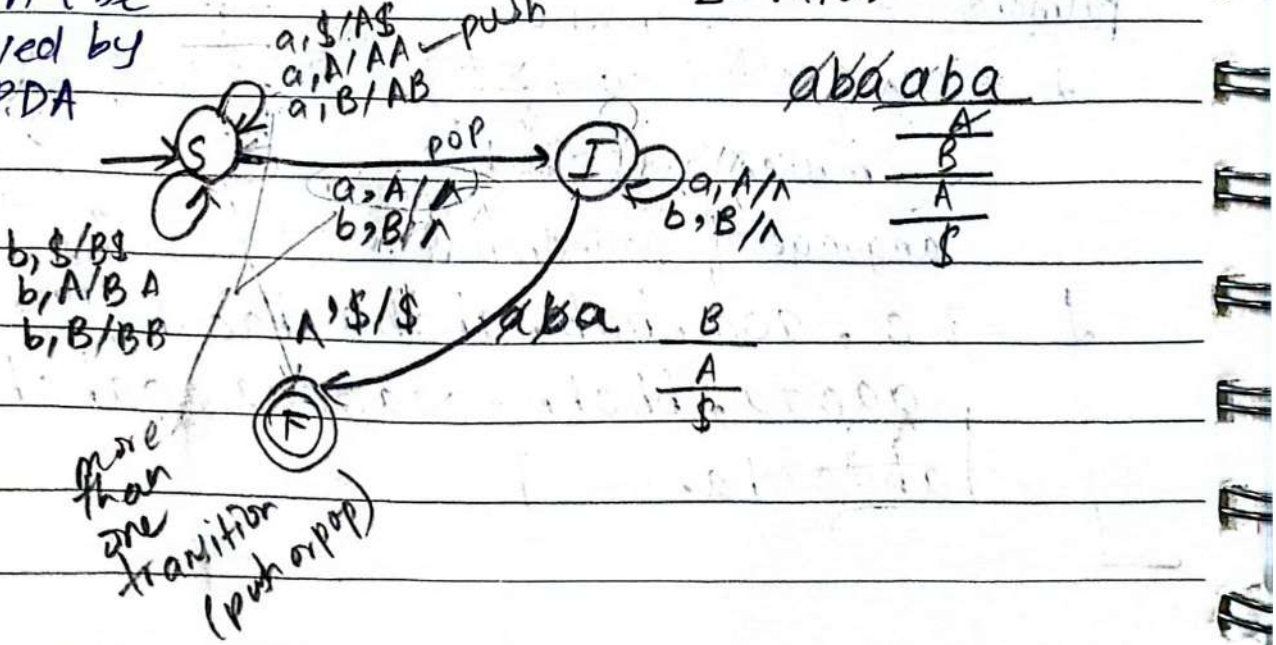
$Q \times \Sigma \times \Gamma \rightarrow \text{subset}(Q \times \Gamma^*)$

↓ all transitions or can be skipped.

$L = \{ WcW^R \; ; \; W \in \{a,b\}^* \}$

can't be solved by DPDA

$\Sigma = \{a, b\}$



31/10/24 **NPDA:** → multiple copies of stack (not a single stack)

1. $L = \{ ww^R \; ; \; w \in \{a,b\}^* \}$ Palindrome.
2. $L = \{ a^i b^j \; ; \; i \le j \le \frac{3i}{2} \}$ $\lceil 1.5 \rceil = 2$
3. $L = \{ a^i b^j \; ; \; \frac{i}{2} \le j \le \frac{3i}{2} \}$
4. $L = \{ a^i b^j \; ; \; i \le j \le 2i \}$

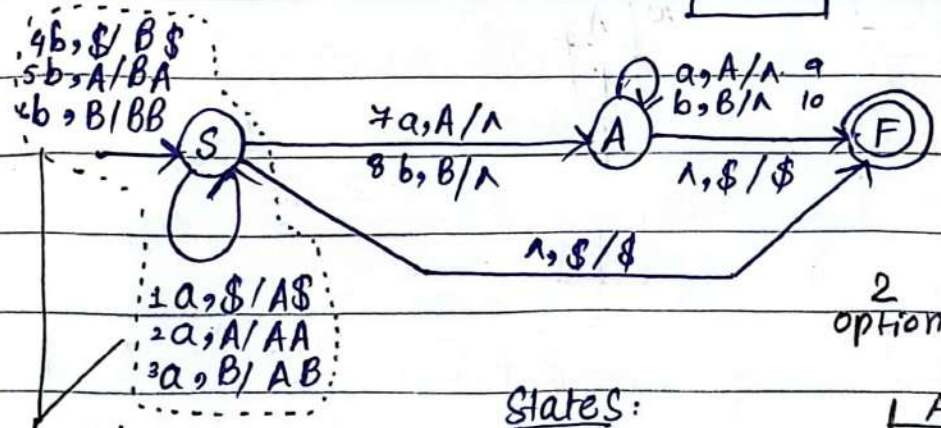
Question#1:

$|ww^R|$ smallest word not odd like a or b.

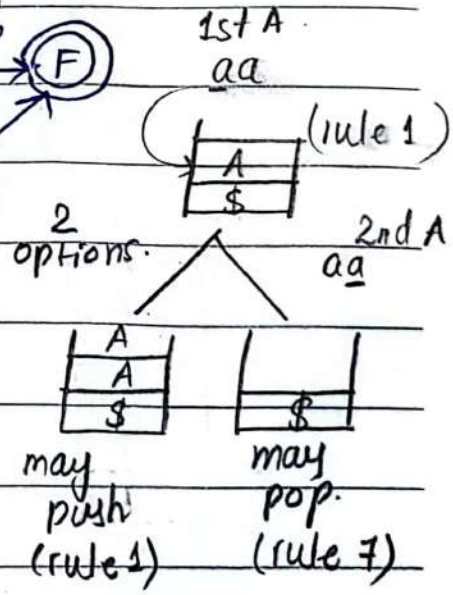
$L = \{ \lambda, aa, bb, abba, baab, \ldots \}$ because of w and w^R (reverse)

$\Gamma = \{ \$, A, B \}$ → 6 combinations there are of even length

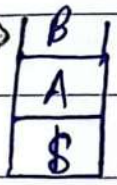
$\Sigma = \{a, b\}$



push (6 transitions)



States:



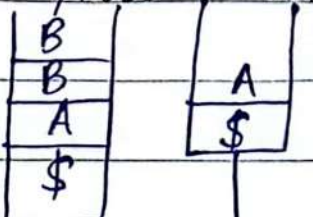
rule 5

$x = abba$

Deterministic transition. (rule 1)

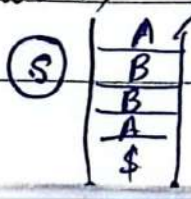
abba

rule 6 rule 8



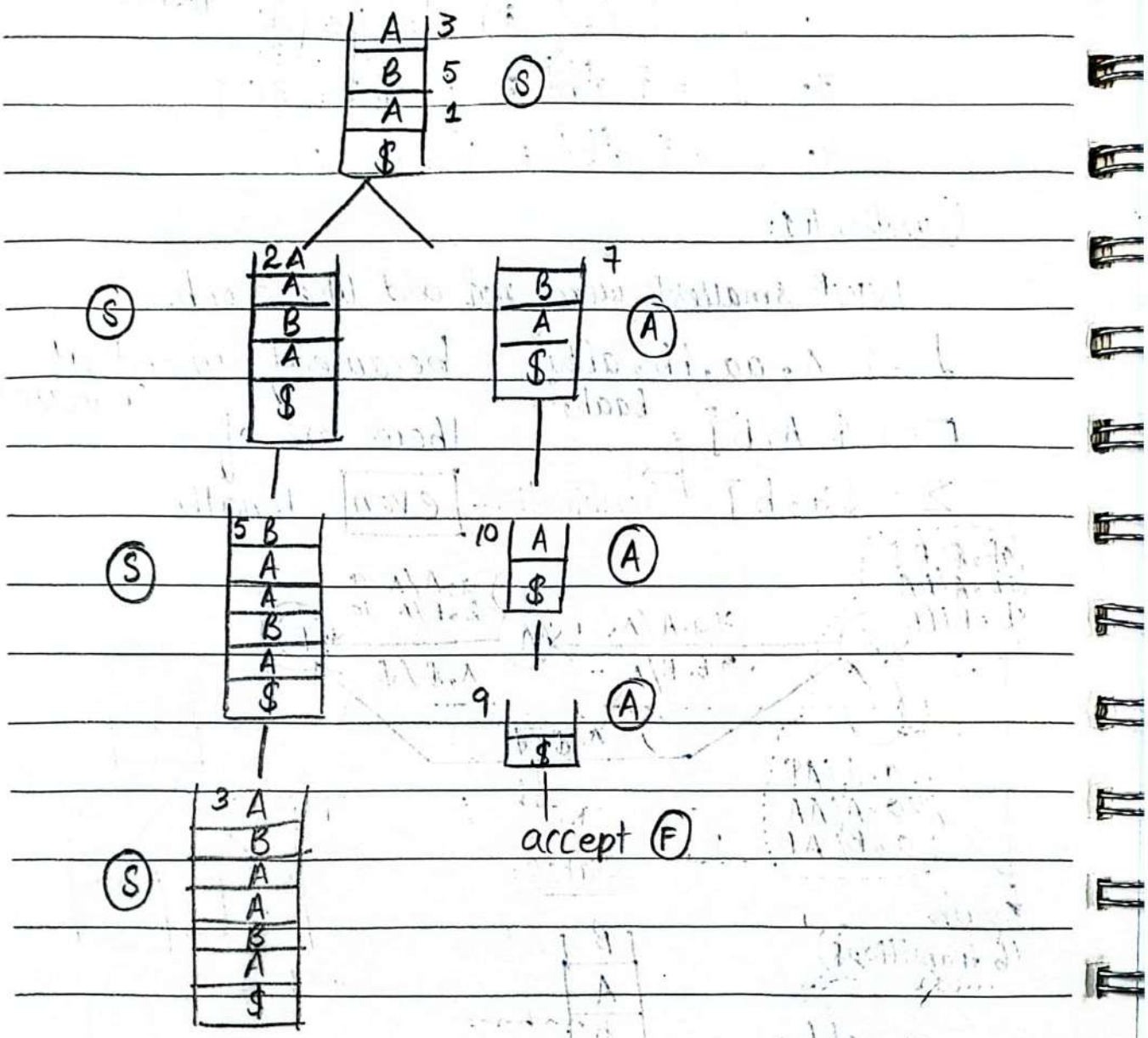
A state. abba

(rule 3) (rule 9)



then accept.

x = abaaba



x = aaaa

5) $L = \{ a^i b^j \; ; \; i \le j \le 2i \}$ → 1.5 → ceil (2)

6) $L = \{ a^i b^j \; ; \; i \le j \le \frac{3}{2} i \}$

7) $L = \{ a^i b^j \; ; \; \frac{i}{2} \le j \le \frac{3i}{2} \}$

→ ceil. / floor

→ mathematically: $0.5i$ (ceil (1)) ... $1.5i$ (ceil (2))

all three question same no

$1i \le j \le 2i$

Q#	
	$i \le j < 2i$
	$i < j < 2i$
	$i \le j \le 2i$

** when more than one condition then NDPA

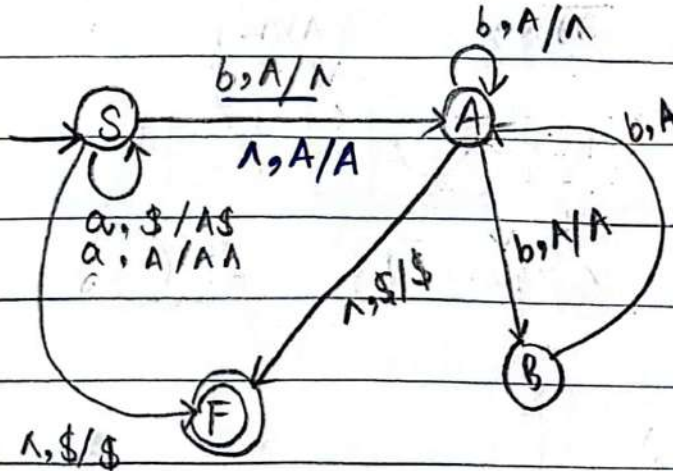
$i=j$ $i<j$

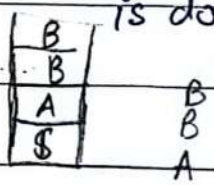
$L = \{ a^i b^j \; ; \; i \le j \le 2i \}$

$S \longrightarrow \wedge / aSb / aSbb$

	i	j	$2i$
both null ←	0	0	($\wedge$) 0
$i=j$ (equal)	1	1 2	2
	2	2 3 4	4
	3	3 4 5 6	6

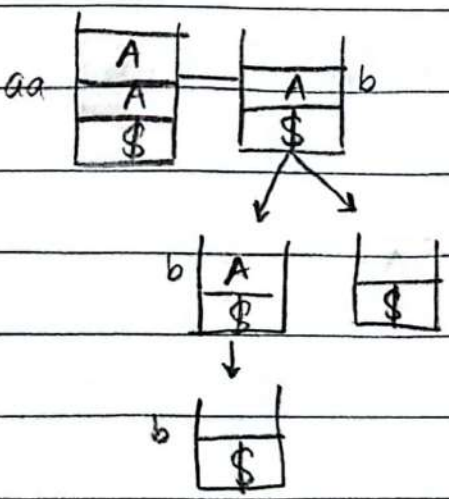
↓ max is double.



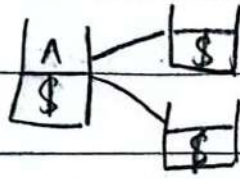


$x = aabbb$

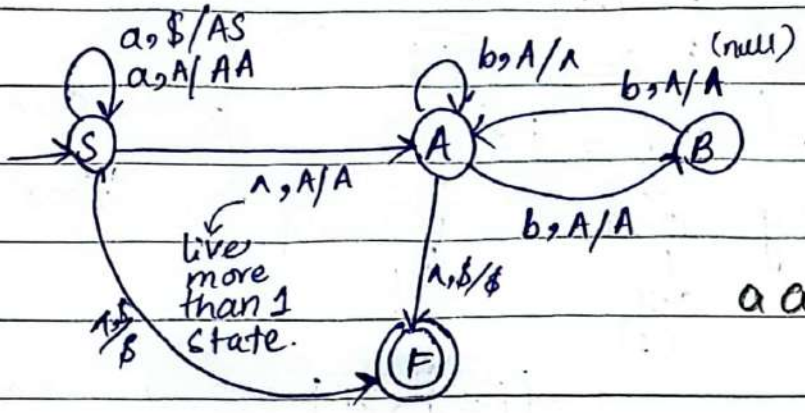
<2 = can't hold



$x = abb$

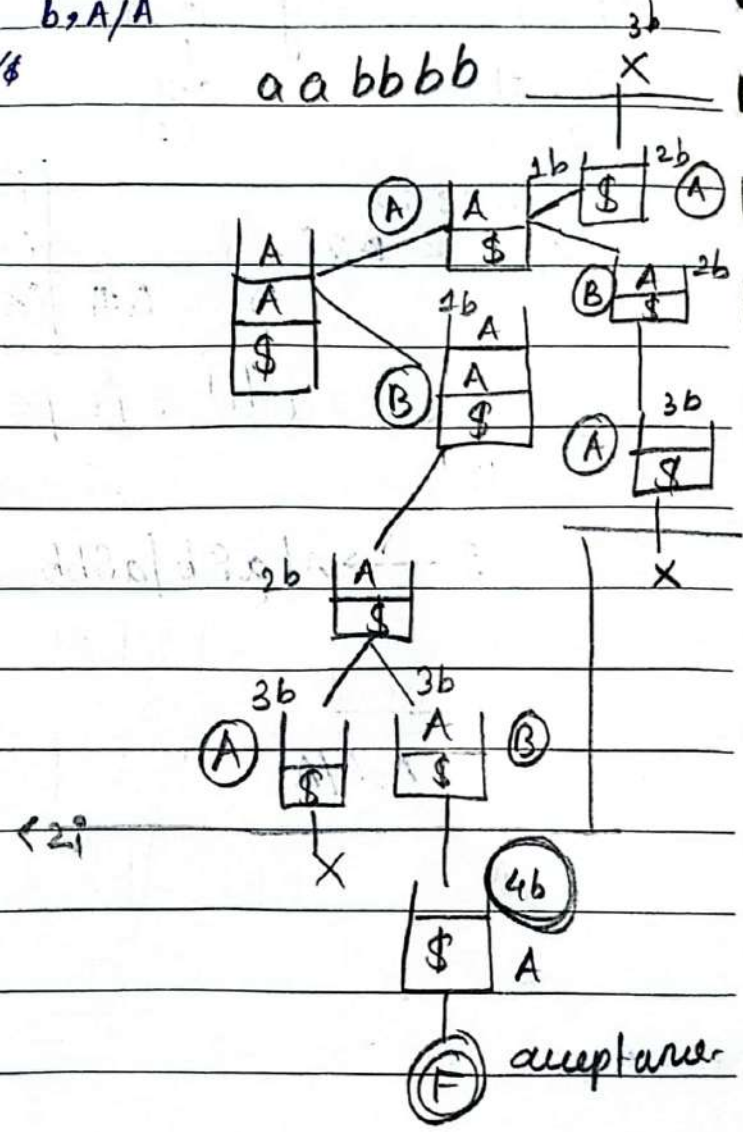


↓ to read second b we need A so error



S-7
Q# SICIP

aabbbb



→ AND $M_1 \cap M_2$ $(F_1 \times F_2)$

→ OR $M_2 \cup M_1$ $\{F_1 \times Q_2\} \cup \{F_2 \times Q_1\}$

NFA → DFA Subset construction method
↓ Transition Table.

0—N comb $(a+b+\lambda)^N$

only N $(a+b)^N$

$(a+b+c)^* \rightarrow (a^* b^* \cdot c^*)^*$

DFA → RE State elimination method.

NFA-λ extended transition function

$$\delta^*(A,a) = \lambda\left(\bigcup \delta(K,a) \mid K \in [\delta^*(A,\lambda)]\right]$$

(new ch: a; prev arg: $\delta^*(A,\lambda)$)

$$= \lambda(\delta(A,a) \cup \delta(B,a) \cup \delta(C,a)]$$

$$= \lambda([A] \cup \phi \cup \phi])$$

$$= \lambda([A]) = [A,B,C]$$

DFA:

$$\delta^*(SI,abb) = \delta(\delta(\delta(\delta^*(SI,\lambda),a),b),b)$$

NFA-λ → NFA Null (λ) elimination method / Kleene's theorem.

⇒ $(\lambda, \Sigma, \lambda)$

RE → NFA-λ

CFG → CNF

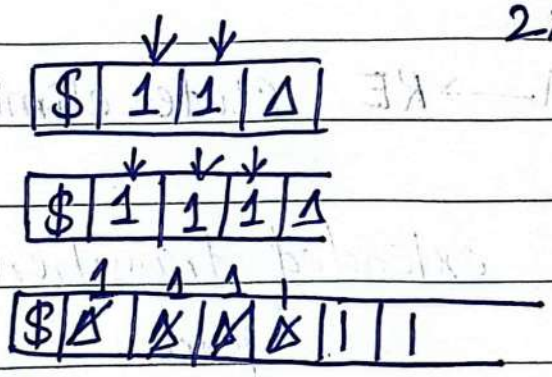
CNUSM

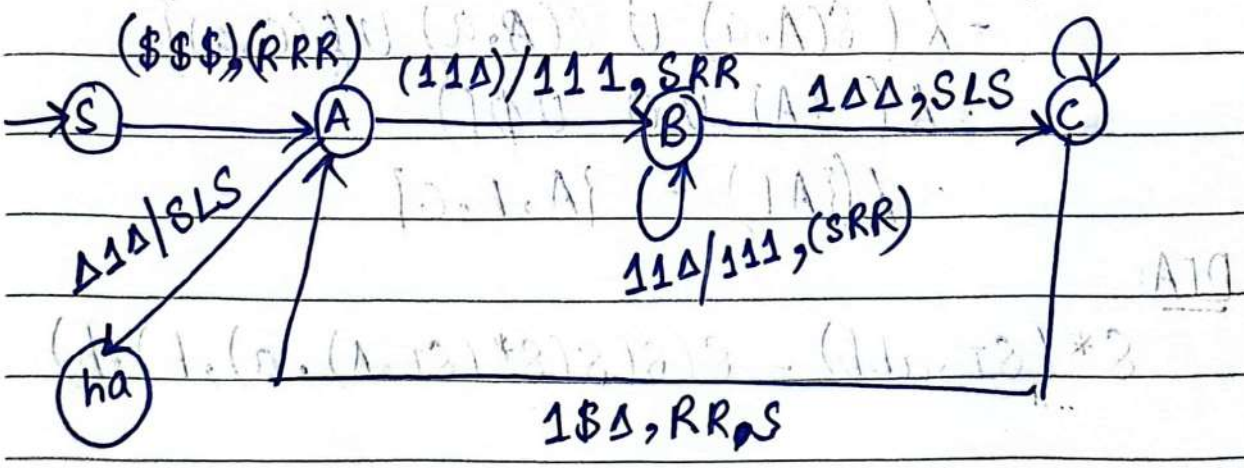
multitape turing machine :- MTM

$\Sigma_{TM1}\ \Sigma_{TM2} / \Sigma_{TM1}\ \Sigma_{TM2}, D_1 D_2$

read write

Three Tapes = 2×3





non-context free ← TOA → CF-L → RL

12/11/24

Turing Machine → FA + Tape

$W \in \{a,b\}^*$ | NCW | $a^n b^n c^n$

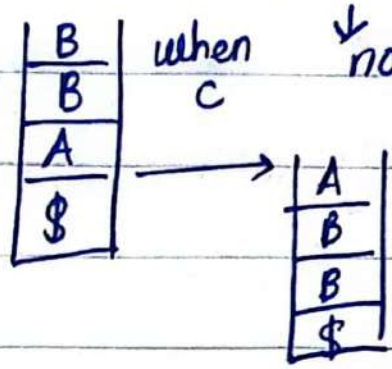
abbcabb | more than two variables

Δ (blank symbol)

$	a	b	b	c	a	b	b	Δ	Δ
..	Δ	Δ	Δ	...	Δ	Δ	Δ	...	
								Δ	

movements allowed.

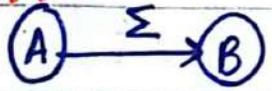
- → read
- → write
- → modify



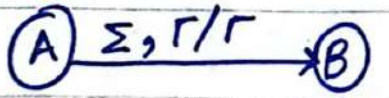
use stack 2 now.

* first input tape was not modifiable, and unidirectional in PDA, DFA, NFA, NFA-λ. Only used for string processing.

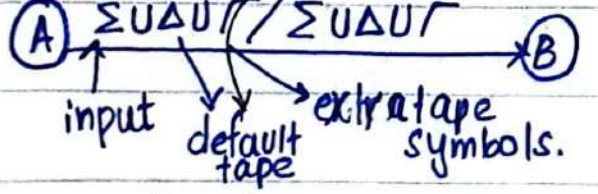
FA:



PDA:



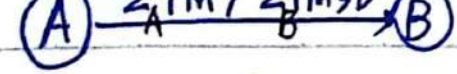
TM: → U $ (tape start) → one end of tape is closed ($) → tape + pointer



$\Sigma_{TM} = \Sigma \cup \Delta \cup \Gamma \cup \$$

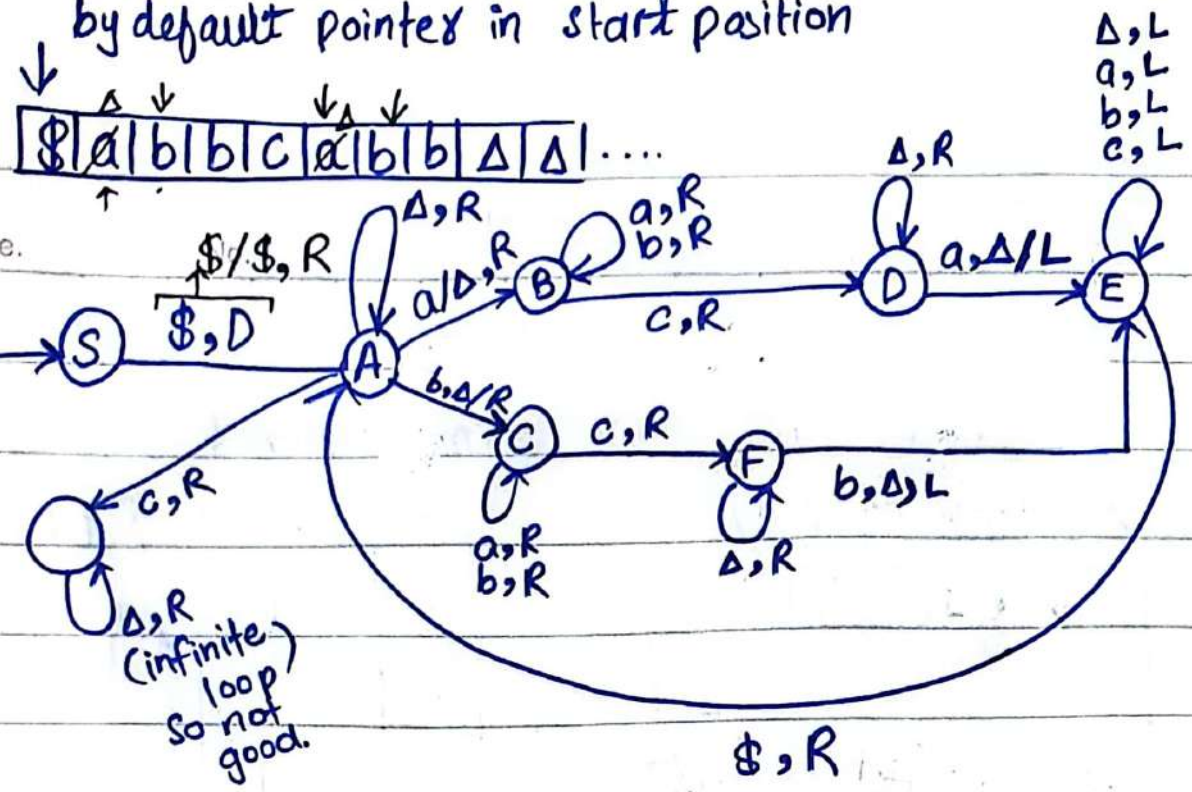
(read) / (write)

(movements) direction = {L, R, S} left, right, static



if $\Sigma_{A\,TM} = \Sigma_{B\,TM}$ then simply write Σ_{TM}, D

by default pointer in start position



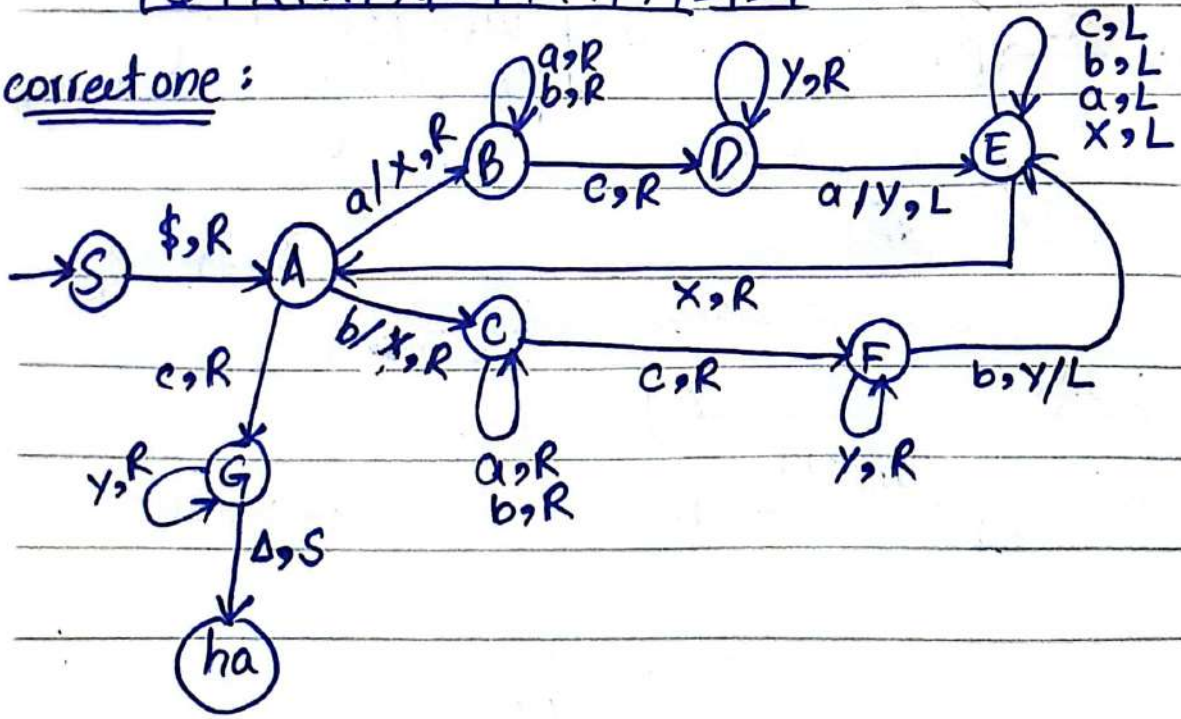
So, use X instead of Δ

$	X	X	X	c	X	X	X	Δ	Δ	....

use Y instead of X after C

$	X	X	X	c	Y	Y	Y	Δ	Δ	.....

correct one :

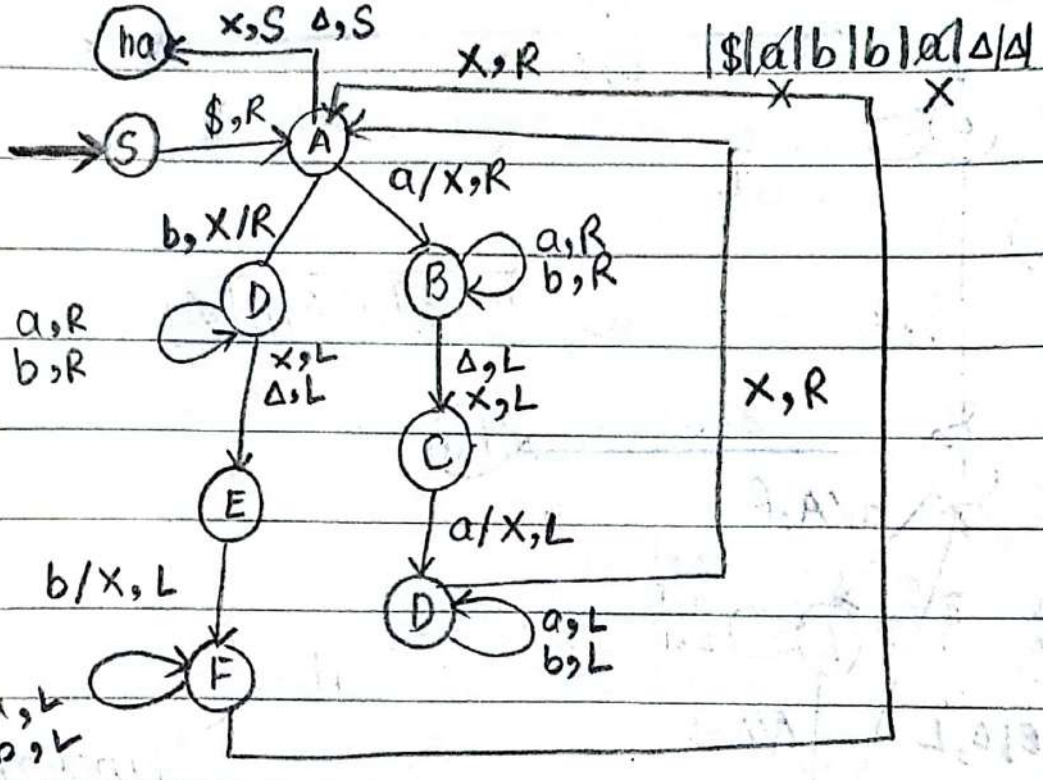


Turing machine:

1) $L = \{ ww^R \mid w \in \Sigma^* \}$ Date. 14/11/24

$\Sigma = \{a, b\}$

$L = \{ \lambda, aa, bb, abba, baab, \ldots \}$



2) $L = \{ WW \mid W \in \Sigma^* \}$ $\Sigma = \{a, b\}$

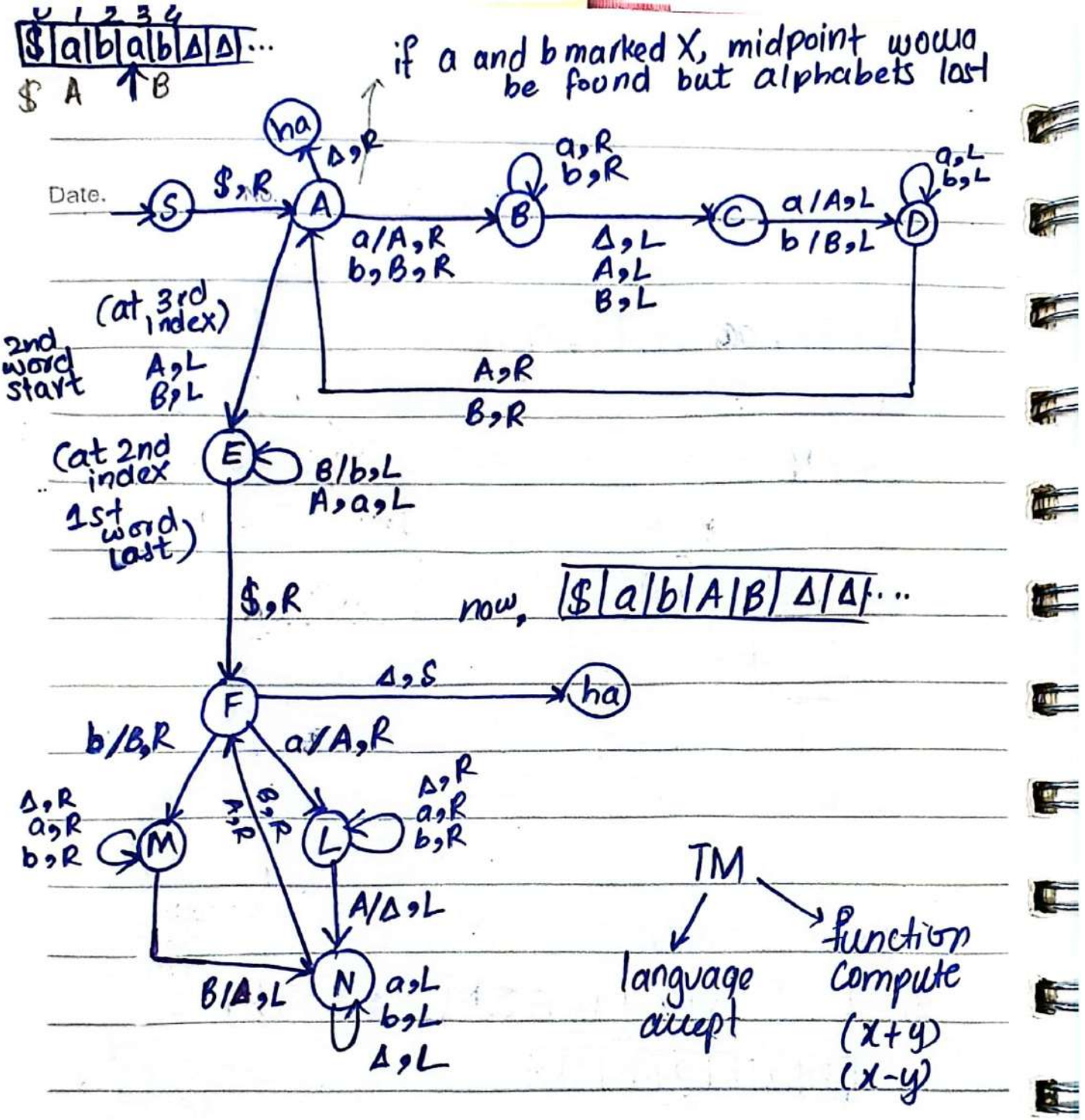
0	1	2	3	4	5	
$	a	b	a	b	Δ	...

↑

→ only even length Scenario will work

method 1: 3rd index X

$ next Y


0 1 2 3 4
$ a b a b Δ Δ ...
$ A B
if a and b marked X, midpoint wowa be found but alphabets last
ha
Δ,R
$,R
S
A
a/A,R
b,B,R
B
a,R
b,R
Δ,L
A,L
B,L
C
a/A,L
b/B,L
D
a,L
b,L
A,R
B,R
(at 3rd index)
2nd word start
A,L
B,L
E
(at 2nd index 1st word last)
B/b,L
A,a,L
$,R
now, $ a b A B Δ Δ ...
F
Δ,S
ha
b/B,R
a/A,R
M
Δ,R
a,R
b,R
A,R
B,R
L
Δ,R
a,R
b,R
A/Δ,L
N
B/A,L
a,L
b,L
Δ,L
TM
language accept
function compute
(x+y)
(x-y)

19/11/24 Turing Machine

Date. No.

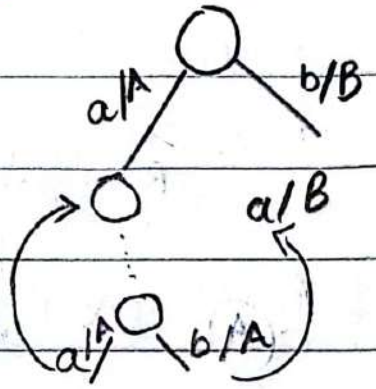
L = { W | W is Reverse of W ; using same memory }

input: B → A

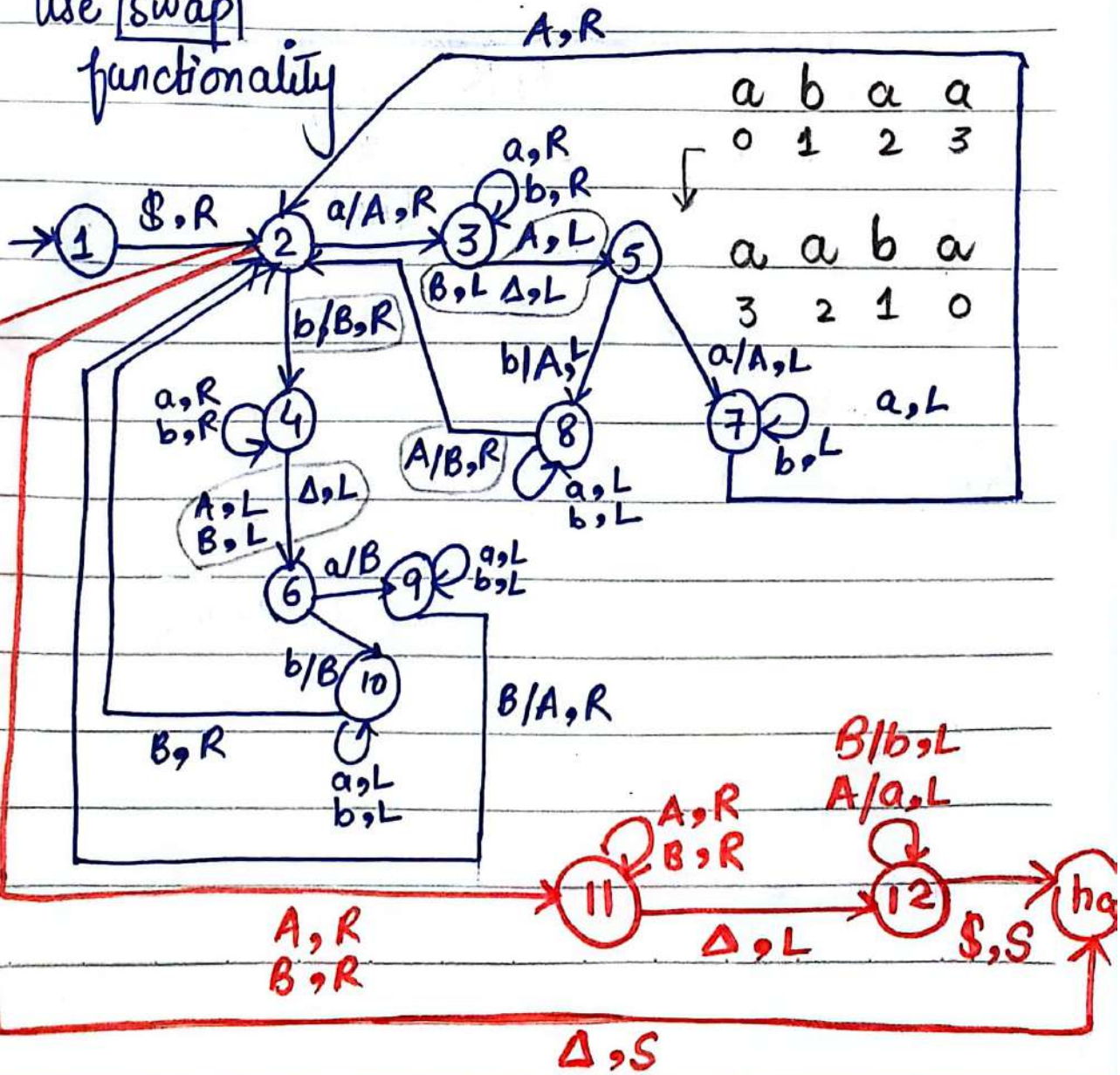
| $ | a | b | a | b | Δ | Δ | Δ | ... |

output: A B B

| $ | b | a | b | a | Δ | Δ | Δ | ... |



use Swap functionality



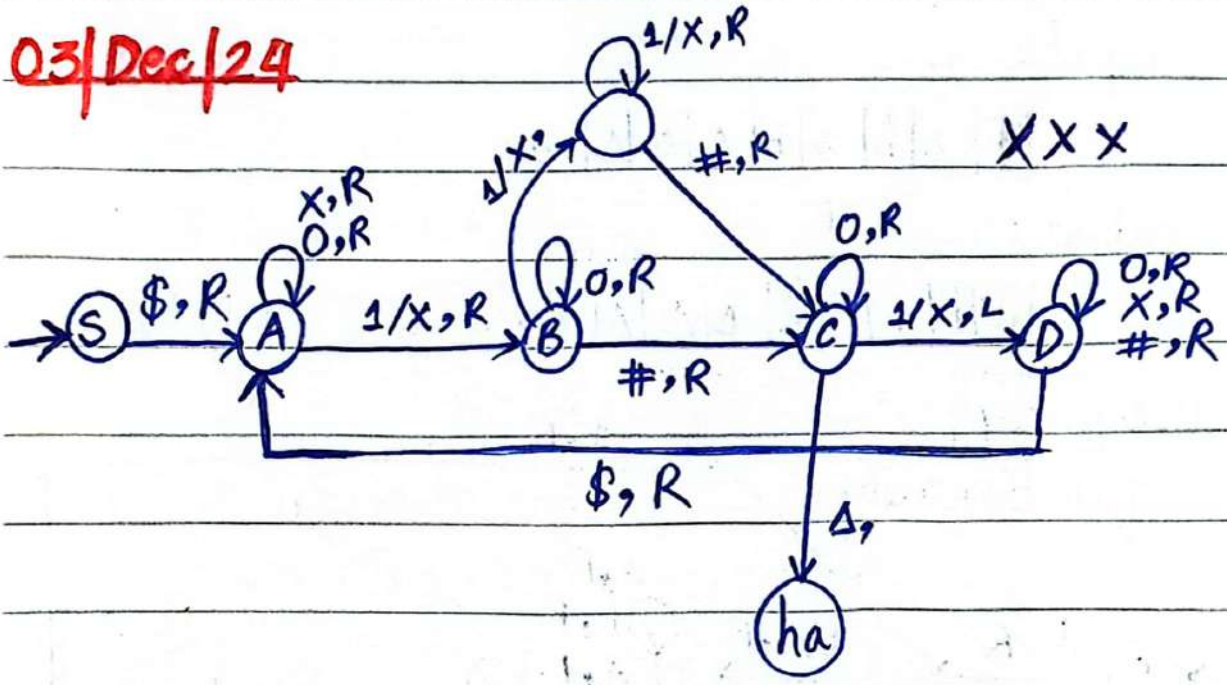
Date. No.

error

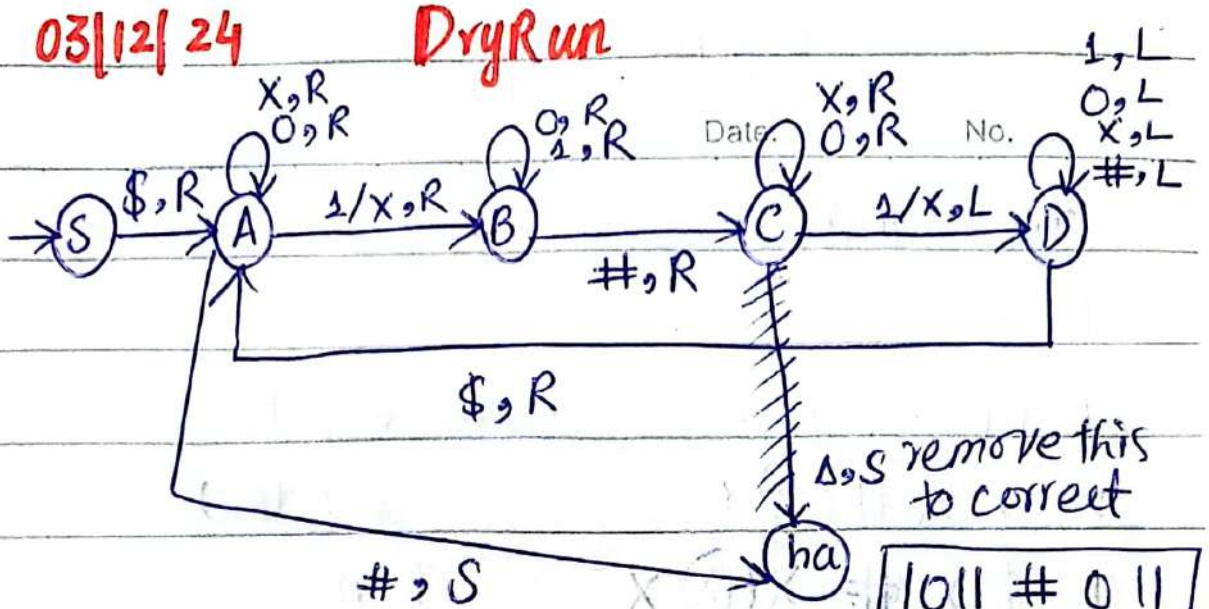
$q_5 \rightarrow q_8$

236: input processing

03/Dec/24



03|12|24 DryRun



Input : |$|1|0|1|#|0|1|1|Δ|

Output : |$|X|0|1|#|0|X|1|Δ|

|$|X|0|1|#|0|X|1|Δ|

|$|X|0|X|#|0|X|X|Δ|

|$|X|0|X|#|0|X|X|Δ|

accepted by A to ha

⇒ Count if #of 1's in first word before # is equal to # of 1's after #.

Dry run: $ 1 # 01

Date. No.

⊢ (S) $1 #01

⊢ $ (A) 1 #01

⊢ $ X (B) #01

⊢ $ X # (C) 01

⊢ * $ X # 0 (C) 1

⊢ $ X # (D) 0 X

⊢ * (D) $ X # 0 X

⊢ $ (A) X # 0 X

⊢ * $ X (A) # 0 X

⊢ $ X (ha) # 0 X

e.g if
$ 1 # 0 1

Then
⊢ $ X # (C) 01

⊢ * $ X # 0 (C) 1
↓
skip all things in loop

e.g
⊢ $ X # (C) 000 1

⊢ * $ X # 000 (C) 1

Date. No.

a	b	a	#	a	b	a	b	a	Δ
				A	B	A	B	A	

read write.

$\Sigma_{T1}\ \Sigma_{T2} / \Sigma_{T1}\ \Sigma_{T2}, D_1 D_2$

5/04/24 1) Multitape Turing Machine

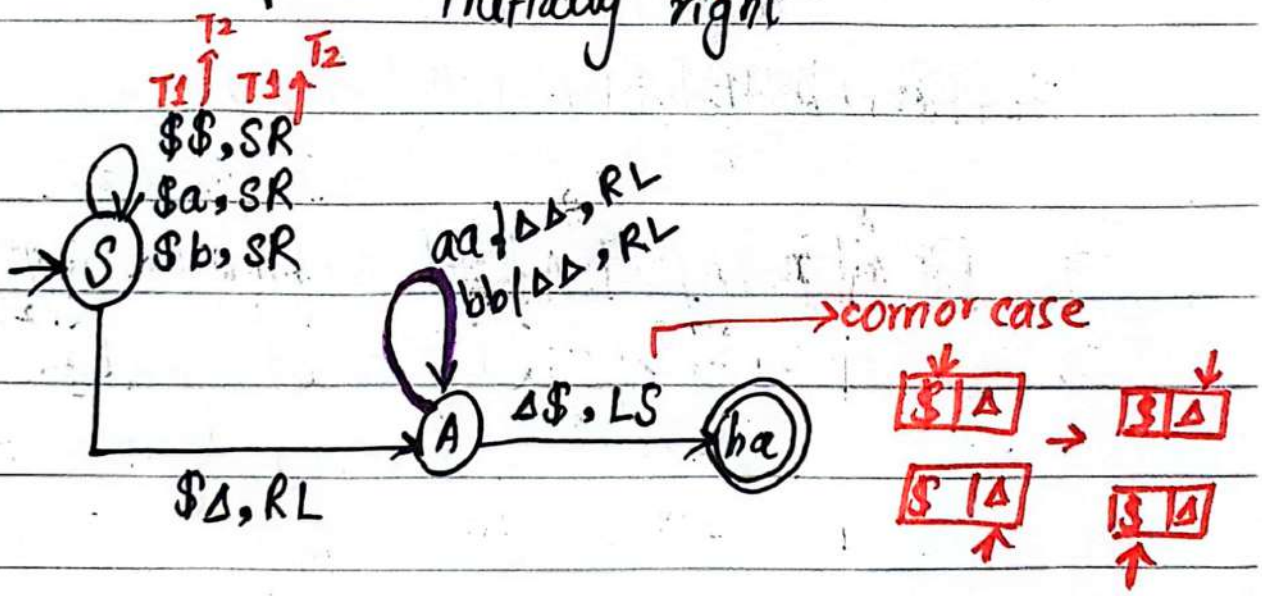
2) Multitrack

initially static

W | $ | a | a | b | a | b | Δ | Tape #1

W^R | $ | b | a | b | a | a | Δ | Tape #2

initially right



▲ Multitape turing machine.

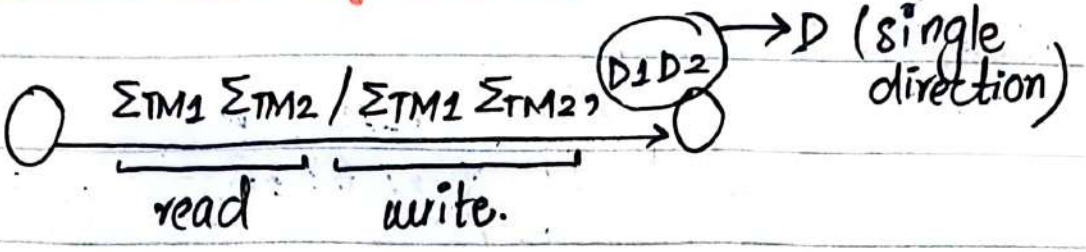
highest complexity

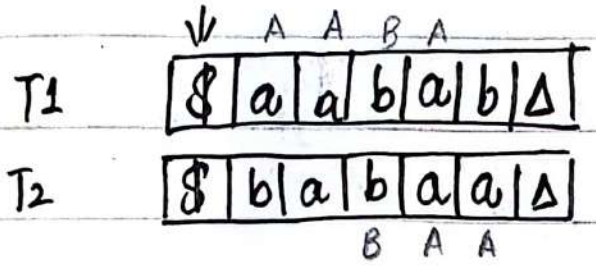
1) Multitrack
2) Single Tape
3) Multi Tape.

(same direction)

Multitrack Turing Machine

Date: No.

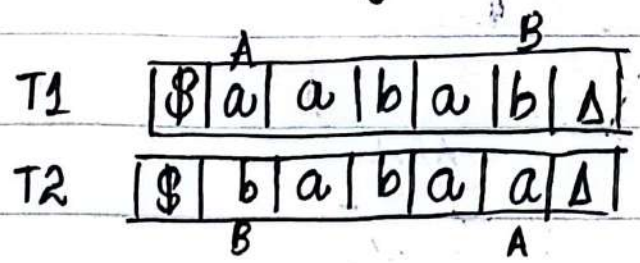




* approach should be very optimised to avoid complexity.

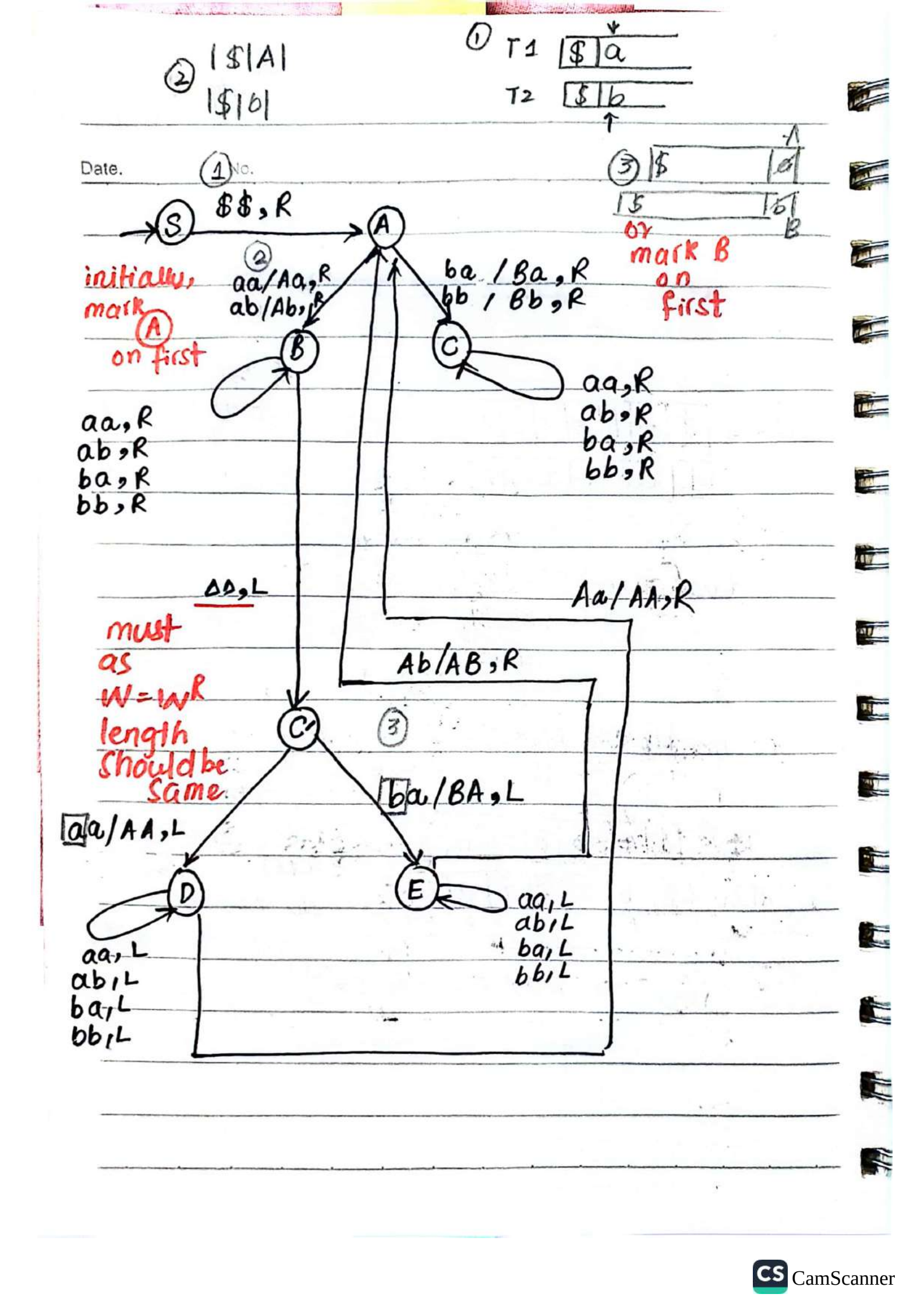
4 variables $2^4 = 16$ transitions possible in one State.

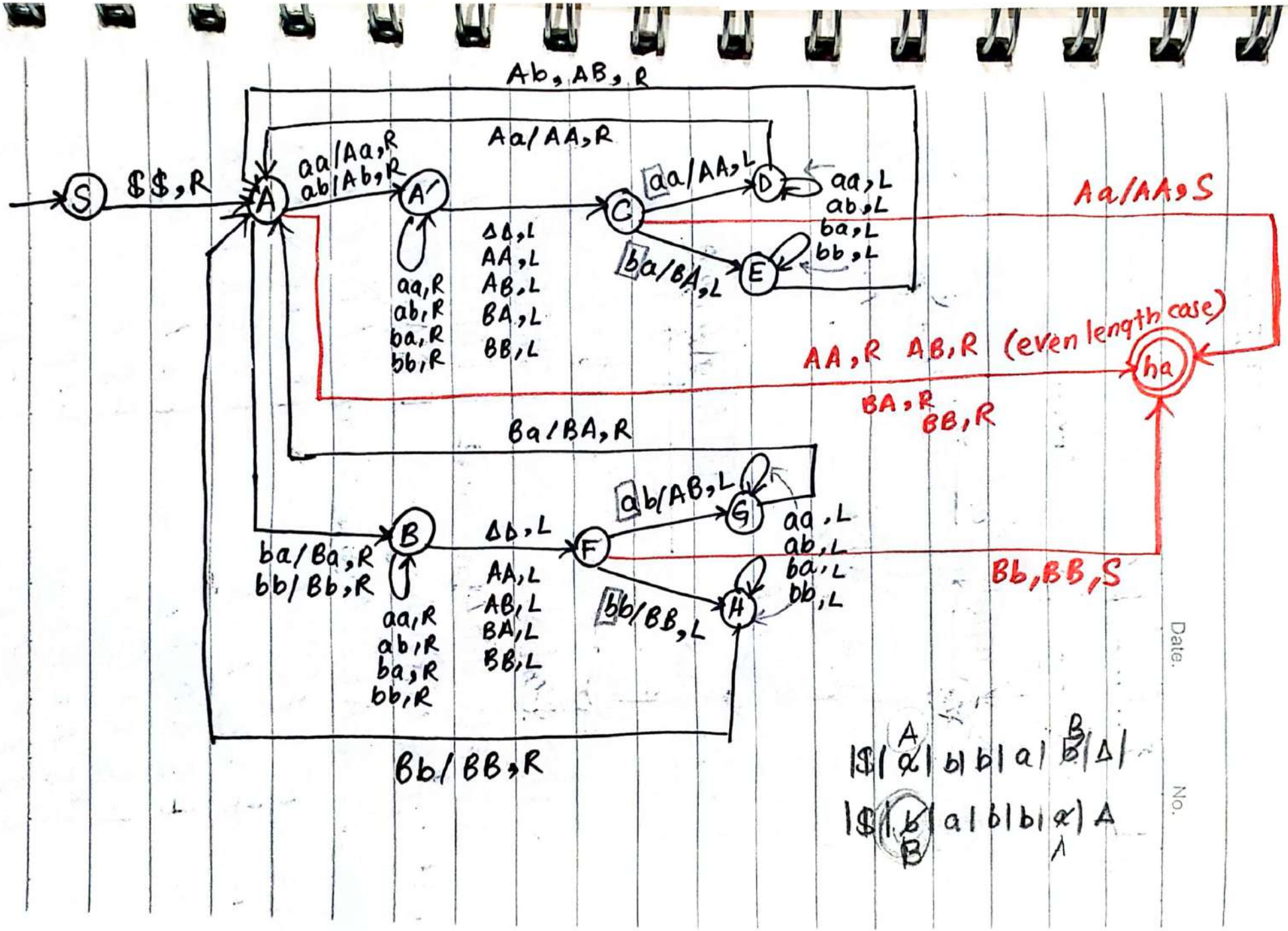
Optimal Possible Algorithm:-



as you mask first and last (T1) (T2) do vice versa simultaneously

So, now max transitions = 4

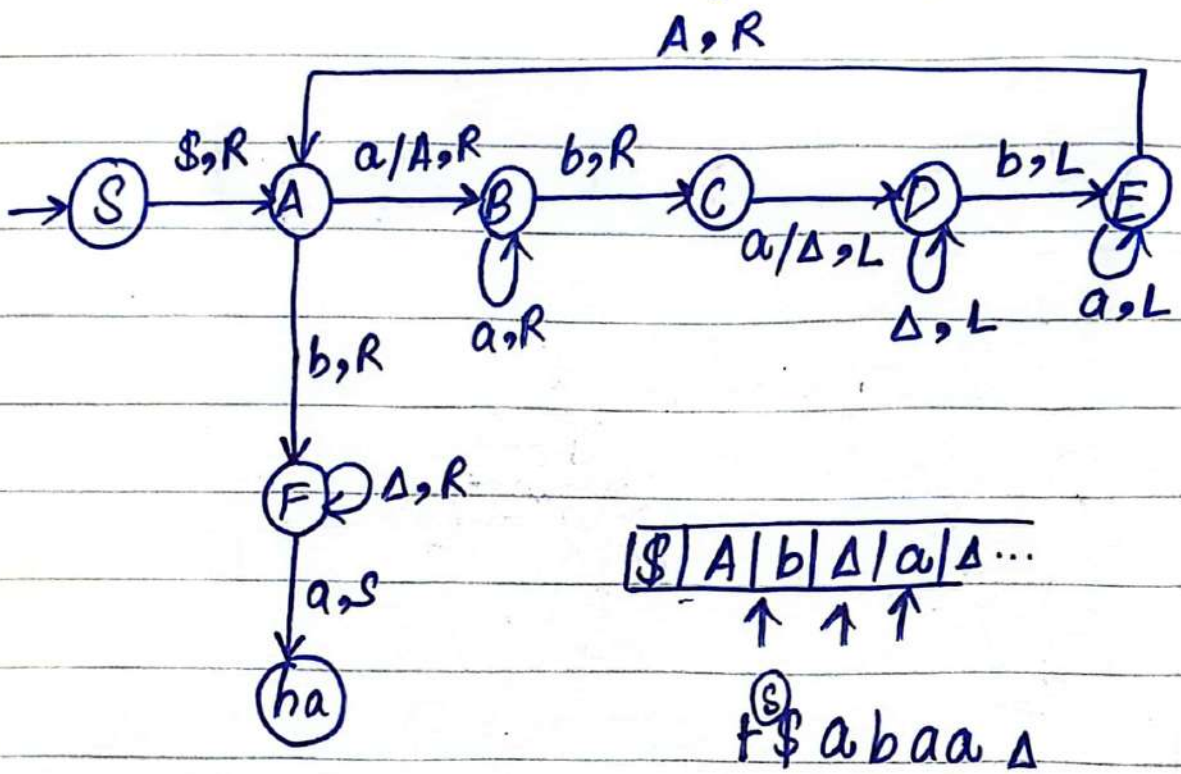

① T1 $ a
T2 $ b
② |$|A|
|$|0|
③ $ a
$ b
B
or mark B on first
Date.
1
No.
$$,R
S
A
initially, mark A on first
aa/Aa,R
ab/Ab,R
ba/Ba,R
bb/Bb,R
B
C
aa,R
ab,R
ba,R
bb,R
aa,R
ab,R
ba,R
bb,R
ΔΔ,L
must as W=WR length Should be same
Aa/AA,R
Ab/AB,R
C
3
ba/BA,L
aa/AA,L
D
E
aa,L
ab,L
ba,L
bb,L
aa,L
ab,L
ba,L
bb,L


Ab, AB, R
Aa/AA, R
aa/Aa, R
ab/Ab, R
S
$$, R
A
A'
aa, R
ab, R
ba, R
bb, R
ΔΔ, L
AA, L
AB, L
BA, L
BB, L
C
aa/AA, L
D
aa, L
ab, L
ba, L
bb, L
ba/BA, L
E
Aa/AA, S
AA, R AB, R (even length case)
ha
BA, R
BB, R
Ba/BA, R
ab/AB, L
G
aa, L
ab, L
ba, L
bb, L
Bb, BB, S
B
ba/Ba, R
bb/Bb, R
aa, R
ab, R
ba, R
bb, R
ΔΔ, L
AA, L
AB, L
BA, L
BB, L
F
bb/BB, L
H
Bb/BB, R
A
B
$ α b b a b Δ
$ b a b b α A
λ

| $ | a | b | a | a | Δ ...

Date. No.

$L = \{ a^i b a^j \mid 0 \le i < j \}$ $j = i+1$



| $ | A | b | Δ | a | Δ ...

↑ ↑ ↑

⊢ (S) $ a b a a Δ

⊢ $ (A) a b a a Δ

⊢ $ A (B) b a a Δ

⊢ $ A b (C) a a Δ

⊢ $ A b Δ (D) a Δ

⊢* $ A (D) b Δ a Δ

⊢ $ (E) A b Δ a Δ

⊢ $ A (A) b Δ a Δ

⊢ $ A b (F) Δ a Δ

⊢* $ A b Δ (F) a Δ

⊢ $ A b Δ (ha) a Δ

(accepted)

⊢ (S) $ a b Δ

⊢ $ (A) a b Δ

⊢ $ A (B) b Δ

⊢ $ A b (C) Δ

⊢* $ A (D) b Δ

⊢ $ (E) A b Δ

⊢ $ A (A) b Δ

⊢ $ A b (F) Δ ...

⊢* $ A b (F) Δ ...

⊢ $ A b (hr) Δ ..

(rejected)