Theory of Automata

Date: Maham

⇒ Week 1: Lecture 1

sobia.tariq@nu.edu.pk

• Set

well defined collection of distinct objects

› finite set of alphabets $\{a \ldots z\}$

› infinite set of integers $\{\cdots -1, 0, \ldots \infty\}$

• Subset

$A \subseteq B$

• Equal

if $A \subseteq B$ and $B \subseteq A$

• Difference

$A - B = \{x \mid x \in A \text{ and } x \notin B\}$

• Union

$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

• Intersection

$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$

• Complement

$\bar{A} = U - A$ Universal set

→ Laws

Commutative law: $A \cup B = B \cup A$

$A \cap B = B \cap A$

Associative law: $(A \cup B) \cup C = A \cup (B \cup C)$

$(A \cap B) \cap C = A \cap (B \cap C)$

Distributive law: $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$

$(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$

eg logically prove

$A \cup (A \cap B) = A$

$A \cap (A \cup B) = A$

Scanned with CamScanner

Date:

- Prefix

abaa

Λ, a, ab, aba, abaa

null concatenated in the end

- Suffix

Language has alphabets

- Concatenation

$$A^+ = \bigcup_{i=1}^{\infty} A^i$$

$$xy \neq yx$$

eg $A = \{a, b\}$

$A^1 = \{a, b\}$

$A^2 = \{a, b\}\{a, b\} = \{aa, ab, ba, bb\}$

$A^3 = \{aa, ab, ba, bb\}\{a, b\}$

$= \{aaa, aba, baa, bba, aab, abb, bab, bbb\}$

$A^+ = \{a, b, aa, ab, ba, bb, aaa, aba, \ldots\}$

Kleene Star

all possible combinations including null

union of A^+ and null, hence min length of alphabet is 0

$A^* = \{\Lambda\} \cup A^+$

⇒ Week 1 : Lecture 2

Enumerate the sets (list down)

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

$L_1 = \{x \mid x \in \Sigma^*$ and x ends with ab $\}$

$L_1 = \{ab, aab, bab, aaab, abab, baab, bbab, \ldots\}$

$L_2 = \{x \mid x \in \Sigma^*$ and x starts with a and ends with b $\}$

$L_2 = \{ab, abb, aab, aaab, abab, aabb, \ldots\}$

Date:

- $L_3 = \{x \mid x \in \Sigma^*$ and x has a substring aab$\}$

  $L_3 = \{$aab, baab, aaab, aaaab, bbaab, ...$\}$ — from start, from end also $\{$aaba, aabb, ...$\}$ $\{$aaaba, baabb...$\}$
- $L_4 = \{x \mid x \in \Sigma^*$ and $|x|$ is even$\}$

  $L_4 = \{\Lambda$, aa, ab, ba, bb, aabb, aaaa, ...$\}$
- $L_5 = \{x \mid x \in \Sigma^*$ and x is a palindrome

  $L_5 = \{\Lambda$, a, b, aa, bb, bab, abba, ...$\}$
- $L_6 = \{x \mid x \in \Sigma^*$ and no. of a's = no. of b's$\}$

  $L_6 = \{\Lambda$, ab, ba, aabb, abab....$\}$
- $L_7 = \{x \mid x \in \Sigma^*$ and x is in the form $a^n b^n a^n$ where $n \geq 0\}$

  $L_7 = \{\Lambda$, aba, $a^2b^2a^2$, $a^3b^3a^3$, ...$\}$
- $L_8 = \{x \mid x \in \Sigma^*$ and $x = a^n$ where n is prime$\}$

  $L_8 = \{$a, aaa, aaaaa, aaaaaaa, ...$\}$

- C++ is a language. If we enumerate all the alphabets we get valid programs

  alphabets (english) $\rightarrow$ words (keywords)

- Finite Automata

  automatic output to input

5 tuple notation

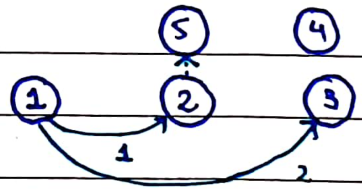
$(Q, \Sigma, A, \delta, q_0)$

- Q → set of states
- Σ → input
- A → set of final state
- δ → transition function
- q_0 → initial state

$\delta\ Q \times \Sigma \rightarrow Q$

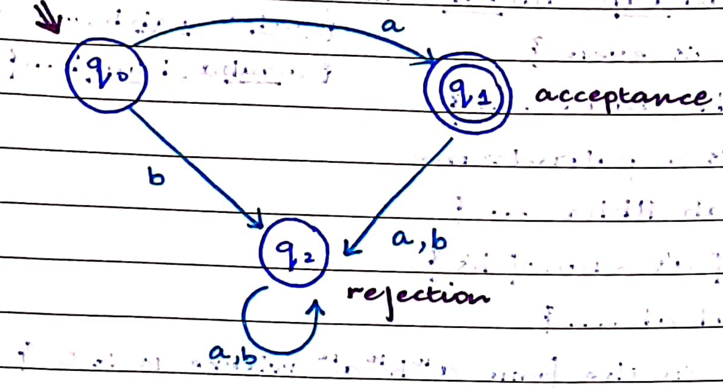
eg $\delta(1,1) \rightarrow 5$

Diagramatic

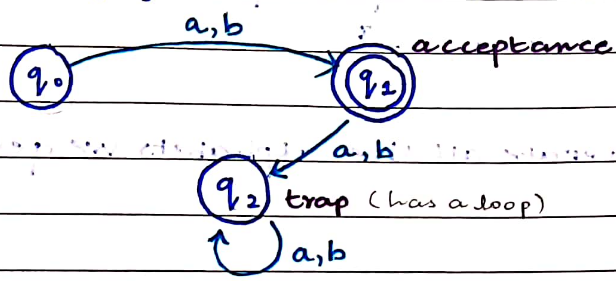


Date:

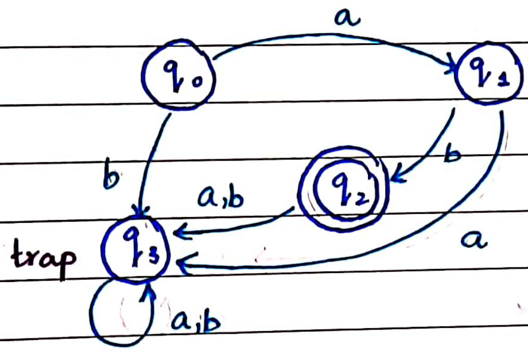
- $L_1 = \{a\}$ $\Sigma = \{a,b\}$



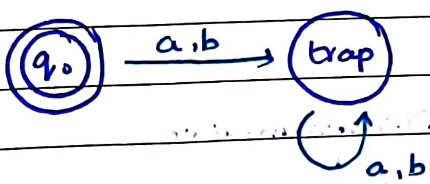
- $L_2 = \{a,b\}$



- $L_3 = \{ab\}$



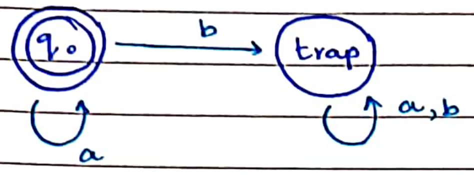
- $L_4 = \{\Lambda\}$



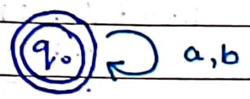
Scanned with CamScanner

Date:

- $L_5 = a^*$



- $L_6 = \{a,b\}^*$

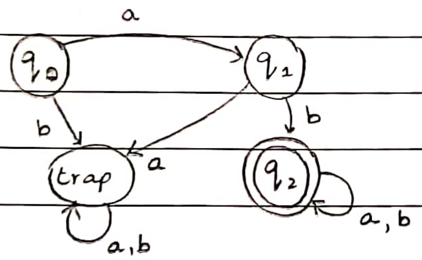


⇒ Week 2, Lecture 1

⇒ Enumerate and form finite automata

- $L_1 = \{x \mid x \in \Sigma^* \text{ and } x \text{ starts with } ab\}$

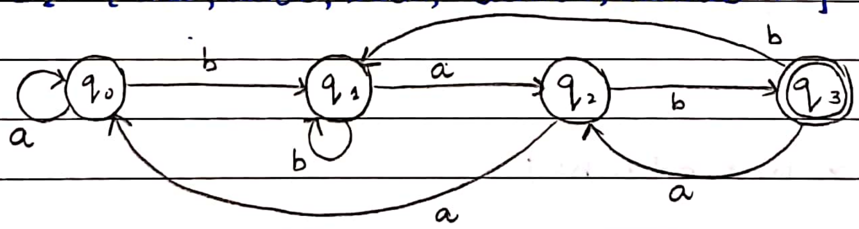
  $L_1 = \{ab, aba, abb, abaa, abab, abba, abbb, \ldots\}$



* partial decessu no trap

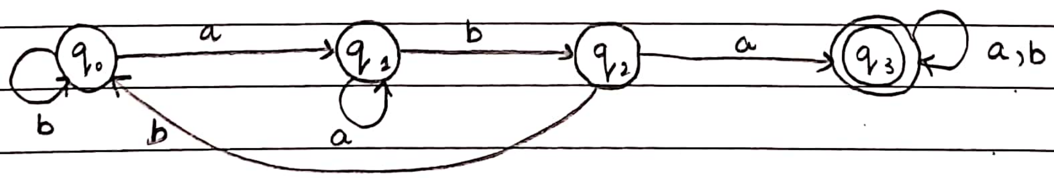
- $L_2 = \{x \mid x \in \Sigma^* \text{ and } x \text{ ends with } bab\}$

  $L_2 = \{bab, abab, bbab, aabab, abbab \ldots\}$



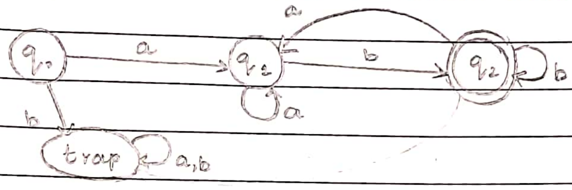
bab
q_1
q_2
q_3

- $L_3 = \{x \mid x \in \Sigma^* \text{ and } x \text{ has substring } aba\}$

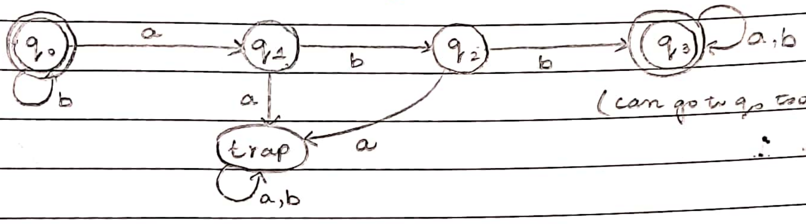


Date:

- $L_4 = \{x \mid x \in \Sigma^*$ and x starts with a and ends with b$\}$



- $L_5 = \{x \mid x \in \Sigma^*$ and every a (if any) in x is followed by atleast 2 b's$\}$

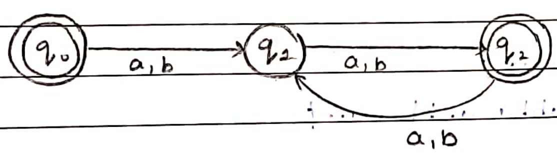


(can go to q_0 too)

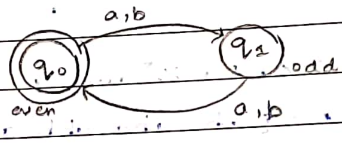
$L_5 = \{\Lambda, b, bb, \ldots$
$abb, abbb, \ldots$
$ab^2 ab^2, ab^3 ab^4, \ldots$

* there is 1 initial state but can be more final states.

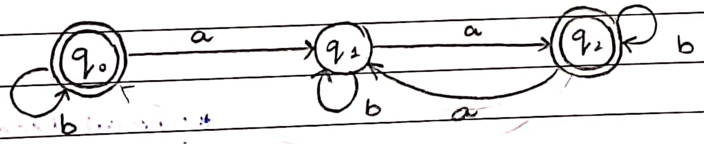
- $L_6 = \{x \mid x \in \Sigma^*$ and $|x| = $ even$\}$

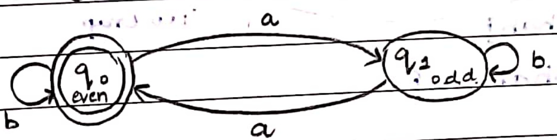


OR

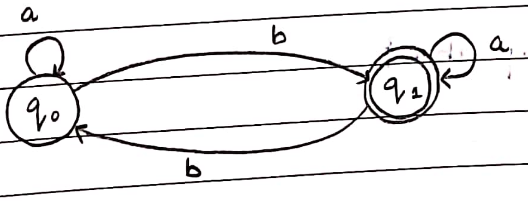


- $L_7 = \{x \mid x \in \Sigma^*$ and x has even a's$\}$





- $L_8 = \{x \mid x \in \Sigma^*$ and x has odd b's$\}$

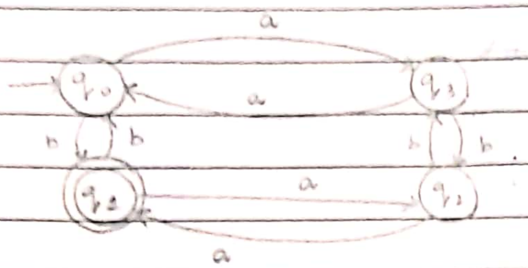


Scanned with CamScanner

Date:

- $L_9 = \{x \mid x \in \Sigma^*$ and x has even a's and odd b's$\}$

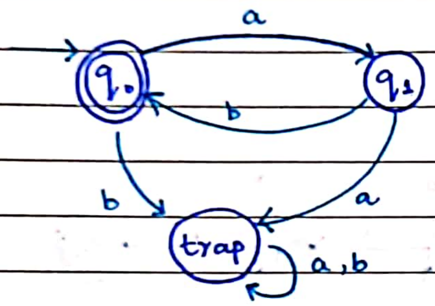
all possible
q_0 > even a's b's
q_1 > odd a's even b's
q_2 > even a's odd b's
q_3 > odd a's, b's



- $L_{10} = \{x \mid x \in \Sigma^*$ and $x = (ab)^*\}$

$L_{10} = \{\Lambda, ab, abab, ababab, \dots\}$

* initial state is final if null accepted



→ Finite Automata

↳ Deterministic FA (at all states, all inputs 1 path)

↳ NFA

↳

⇒ Week 2: Lecture 2

→ Extended function

$(Q, \Sigma, q, A, \delta)$

$\delta \; Q \times \Sigma \to Q$

$\delta^* \; Q \times \Sigma^* \to Q$

Scanned with CamScanner

Date:

1. $\delta^*(q, \Lambda) \rightarrow q$

2. $\delta^*(q, ya) \rightarrow \delta(\delta^*(q, y), a)$

3. $\delta^*(q, xy) \rightarrow \delta^*(\delta^*(q, x), y)$

→ Rule 1,2 Example:

$\delta^*(q_0, ba\underline{a})$ (seperate the right most)

$\delta(\delta^*(q_0, ba), a)$ traverse from left

$\delta(\delta(\delta^*(q_0, b), a), a)$

$\delta(\delta(\delta^*(q_0, \Lambda b), a), a)$

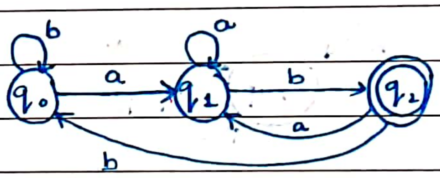
$\delta(\delta(\delta(\underline{\delta^*(q_0, \Lambda)}, b), a), a)$ using rule 1

$\delta(\delta(\delta(q_0, b), a), a)$

$\delta(\delta(q_0, a), a)$

$\delta(q_1, a)$

$q_1 \notin A$ $baa \notin L$



• Traverse baab

$\delta^*(q_0, baab)$

$\delta(\delta^*(q_0, baa), b)$

$\delta(\delta(\delta^*(q_0, ba), a), b)$

$\delta(\delta(\delta(\delta^*(q_0, b), a), a), b)$

$\delta(\delta(\delta(\delta(\delta^*(q_0, \Lambda), b), a), a), b)$

$\delta(\delta(\delta(\delta(q_0, b), a), a), b)$

$\delta(\delta(\delta(q_0, a), a), b)$

$\delta(\delta(q_1, a), b)$

$\delta(q_1, b)$

q_2

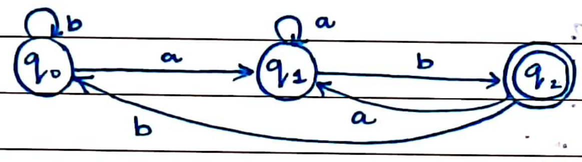
Scanned with CamScanner

Date:

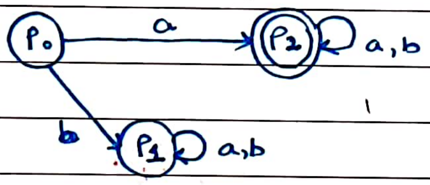
- In rule 3, if left most part already solved we can chunk and solve already solved 2 parts seperately
- → chop down only left part as we have to start from initial state however in abaab, a causes it to start from q_1
- → exception bbaab, here b is self loop so initial state is still q_0 so we can chop

Question

L_1



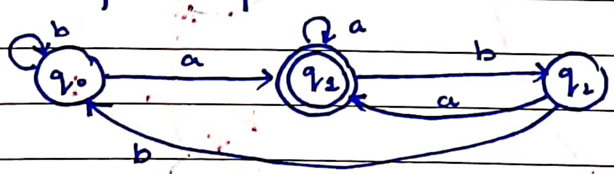
L_2



$L_6 = \{ x \mid x \in \Sigma^* \text{ and } x \text{ doesn't end with } ab \}$

all final states and intermediate states are interchanged in a composite

L_6 is composite of L_1



$L_3 = L_1 \cap L_2$

$= \{ x \in L_1 \text{ and } x \in L_2 \}$

run both, if both accepted then acceptance

$L_4 = L_1 \cup L_2$

$= \{ x \in L_1 \text{ and } x \in L_2 \}$

run both, if one accepted atleast then acceptance

Date:

$L_5 = L_1 - L_2$

$= \{ x \in L_1 \text{ and } x \notin L_2 \}$

Finding finite automata through procedure :

Step 1 : Rename

L1 and L2 should have different state labels

Step 2 : Cartesian product

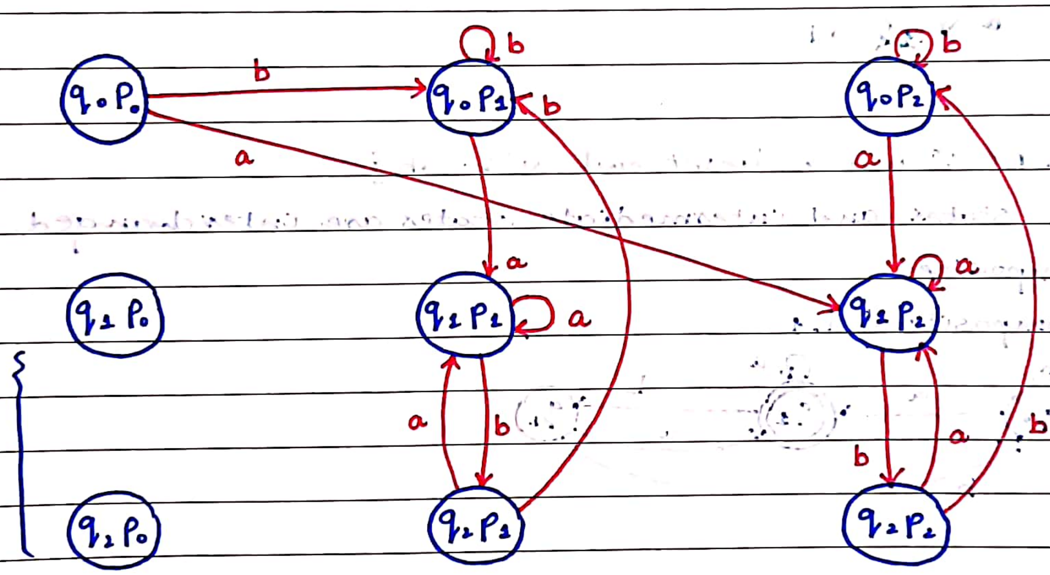
$\{q_0, q_1, q_2\} \times \{p_0, p_1, p_2\}$

$= \{(q_0, p_0), (q_0, p_1), (q_0, p_2)$

$(q_1, p_0), (q_1, p_1), (q_1, p_2)$

$(q_2, p_0), (q_2, p_1), (q_2, p_2)\}$

Step 3 :



these two states are non reachable hence their transitions are not necessary.

- $L1 \cap L2$

$A_3 = \{(q, p) \mid q \in A_1 \text{ and } p \in A_2\}$; $A_1 = \{q_2\}$, $A_2 = \{p_2\}$

$A_3 = \{(q_2, p_2)\}$.

Scanned with CamScanner

Date:

• $L_1 \cup L_2$

$A_4 = \{(q,p) \mid q \in A_1 \text{ or } p \in A_2\}$

$= (q_2, p_1), (q_2, p_2), (q_0, p_2), (q_1, p_2)$

(q_2, p_0) eliminated as isolated state

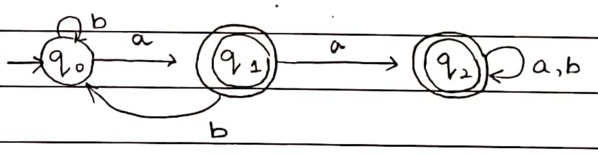
• $L_1 - L_2$

$A_5 = \{(q_2, p_1)\}$

⇒ Week 3: Lecture 1

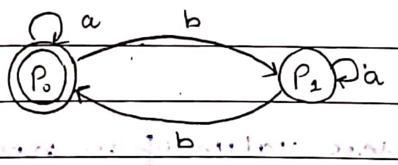
$L_1 = \{x \mid x \in \Sigma^*$ and x has substring aa or ends with a$\}$

$L_2 = \{x \mid x \in \Sigma^*$ and x has even b's$\}$



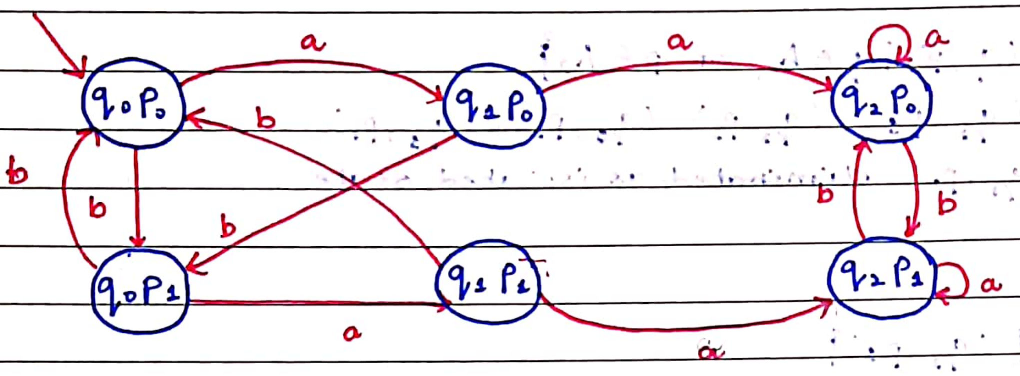
Transition table

	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_2



	a	b
p_0	p_0	p_1
p_1	p_1	p_0

Date:



- $L_1 \cup L_2$

$A_1 = \{q_1, q_2\}$

$A_2 = \{p_0\}$

$L_3 = \{(q,p) \mid q \in A_1 \text{ or } p \in A_2\}$

$= \{(q_1,p_0), (q_1,p_1), (q_2,p_0), (q_2,p_1), (q_0,p_0)\}$

- $L_1 \cap L_2$

$A_5 = \{(q,p) \mid q \in A_1 \text{ and } p \in A_2\}$

$L_4 = \{(q_1,p_0), (q_2,p_0)\}$

- $L_1 - L_2$

$A_5 = \{(q,p) \mid q \in A_1 \text{ and } p \notin A_2\}$

$= \{(q_1,p_1), (q_2,p_1)\}$

circle the final state in the diagram.

- Cannot make finite automata of $a^n b^n$ unless value of n not known. as no. of states should be finite
- Regular languages: finite automata exists
- Non-regular language: finite automata doesn't exist

Scanned with CamScanner

Date:

Simple strings

$|x| = 1$ ∴ element length = 1

Simple language

$L_1 = \{a\}$, $L_2 = \{a, b\}$ ∴ every element length 1

$|\Lambda| = 0$

Regular language

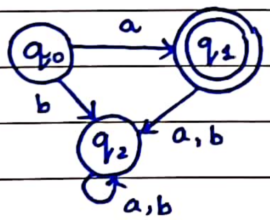
- simple language
- union, intersection, Kleene star, difference, concatenation, complement, reverse of two simple languages.

(simple not necessary, regular for sure)

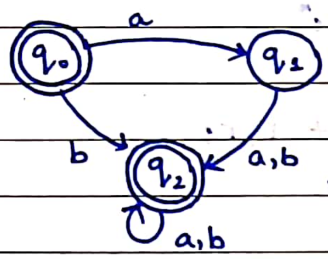
- $L_1 = \{a\}$ $L_2 = \{b\}$
- $L_3 = L_1 \cup L_2 = \{a, b\}$
- $L_4 = L_1 \cap L_2 = \phi$
- $L_5 = L_1^* = \{a\}^*$ Kleene star
- $L_6 = L_1 . L_2 = \{a\}\{b\} = \{a\}\{b\} = \{ab\}$ concatenation
- $L_7 = L_1 - L_2 = \{a\}$ difference
- $L_8 = \bar{L_1}$ = accept everything except a complement

(not simple)

L_1 =



L_8 =



(complement of L_1)

Regular Languages (RL)

Finite Automata → FA

Regular Expression → RE

Date:

Expression

$A = \{0, 2, 4, 6, 8, 10, \ldots\}$

$= 2n$ where $n \geq 0$

$B = \{1, 3, 5, 7, 9, 11, \ldots\}$

$= 2n+1$ where $n \geq 0$

eg $L_1 = \phi$ ϕ

$L_2 = \{\Lambda\}$ Λ

$L_3 = \{a\}$ a

$L_4 = \{a \cup b\}$ $a + b$

$\therefore \{a, b\}$

$L_5 = \{a, aa, aaa, \ldots\}$ a^+

$L_6 = \{\Lambda, a, aa, aaa, \ldots\}$ a^*

$L_7 = \{a$ followed by $b\}$ ab

$L_1 \rightarrow r_1$ $L_2 \rightarrow r_2$

$L_3 = L_1 \cup L_2$

$= r_1 + r_2$

$L_4 = L_1^* = r_1^*$

$L_5 = L_1 L_2$

$= r_1 r_2 \neq r_2 r_1$

$L_6 = L_2 \circ L_1$

$= r_2 r_1$

Find Regular expressions

$L_1 = \{x \mid x \in \Sigma^*$ and x ends with $ab\}$

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

$(a^* + b^*)$ wrong as ab then seperate

$L_2 = \{x \mid x \in \Sigma^*$ and x starts with a and ends with $b\}$

$a(a+b)^* b$

Date:

$L_3 = \{ x \mid x \in \Sigma^*$ and x has a substring ab $\}$

$\therefore (a+b)^* (ab) (a+b)^*$

$L_4 = \{ x \mid x \in \Sigma^*$ and all a's in x (if any) are followed by all b's in x (if any) $\}$

$a^* b^*$

$L_5 = \{ x \mid x \in \Sigma^*$ and every a should be followed by single b $\}$

$b^* (ab)^*$

$L_6 = \{ x \mid x \in \Sigma^*$ and x has exactly 3 a's $\}$

$b^* ab^* ab^* ab^*$

$L_7 = \{ x \mid x \in \Sigma^*$ and x has even a's $\}$

$(b^* ab^* ab^*)^* + b^*$

$L_8 = \{ x \mid x \in \Sigma^*$ and n_a in x are multiples of 3 $\}$

$(b^* ab^* ab^* ab^*)^* + b^*$

→ Week 3 : Lecture 2

Regular Expressions

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

$L_1 = \{ x \mid x \in \Sigma^*$ and $|x|$ is multiple of 4 ...

$((0+1)(0+1)(0+1)(0+1))^*$ 'or' '+' means 1 option chosen

$L_2 = \{ x \mid x \in \Sigma^*$ and every 3^{rd} element in x is 1

$((0+1)(0+1)\,1)^+$ + and not * as Λ excluded

$L_3 = \{ x \mid x \in \Sigma^*$ and 1^{st} two elements are reverse of last 2 $\}$

$00(0+1)^*00 + 11(0+1)^*11 + 01(0+1)^*10 + 10(0+1)^*01$

$00^+ + 11^+ + 101 + 010$

0^+ means atleast 1, 0 will be there $0(1+0)^*0$

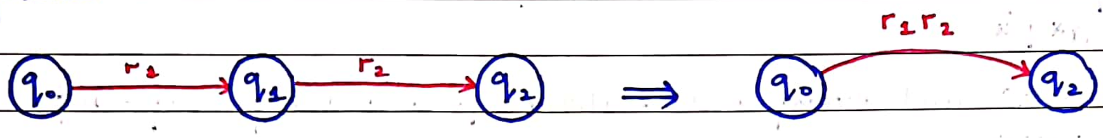
can make wrong eg 0010

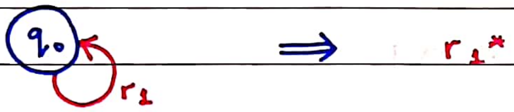
Date:

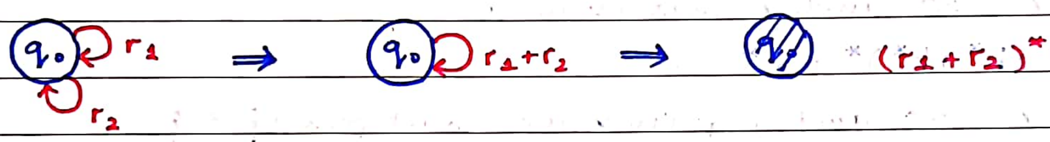
→ State Elimination

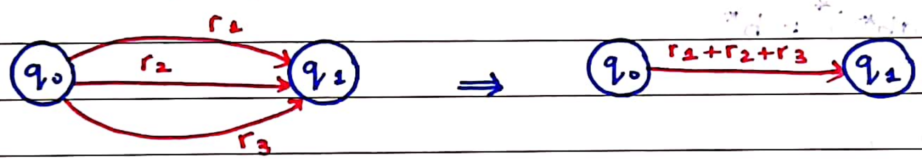
we eliminate all states except initial and final one by one

Rules

1\. 

2\. 

3\. 

eg. 

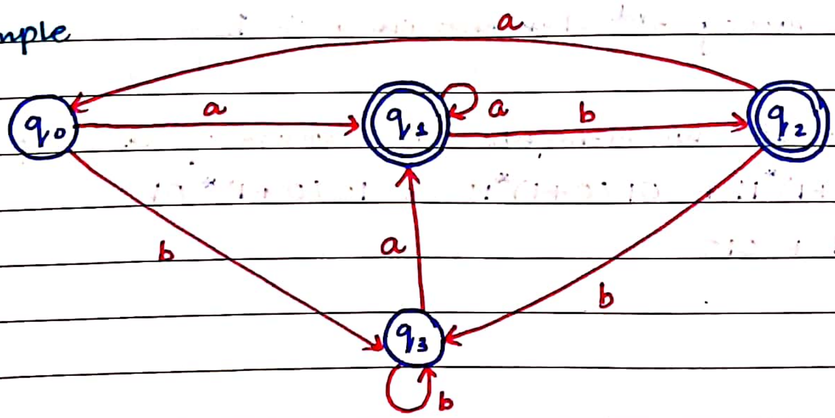
• Note:

Dummy Initial : Initial State → No incoming

Dummy final : final state → No outgoing

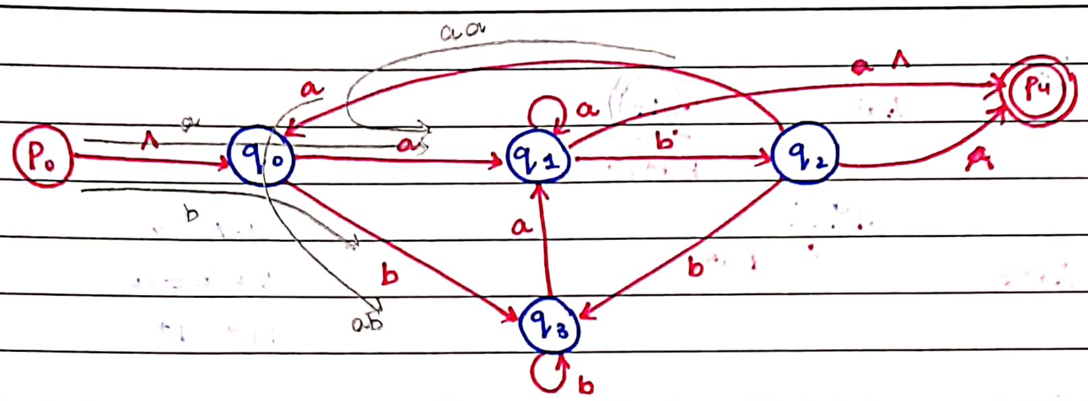
+ only 1 final state

Example



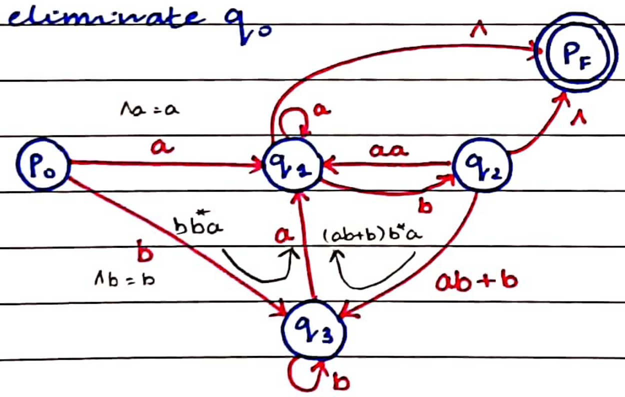
Scanned with CamScanner

Date:



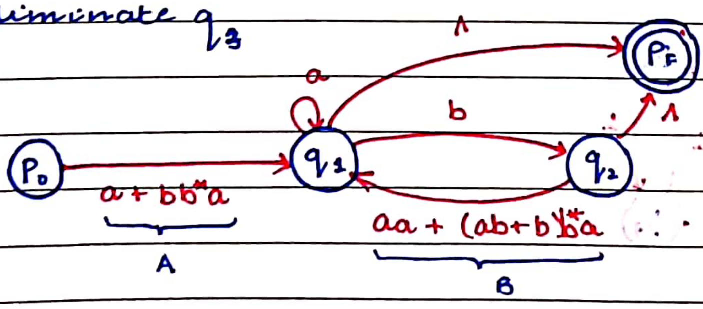
- for a language we can have multiple finite automata and regular expressions.
- The order in which we eliminate states changes regular expressions
- If labelled remove in order

1. eliminate q_0



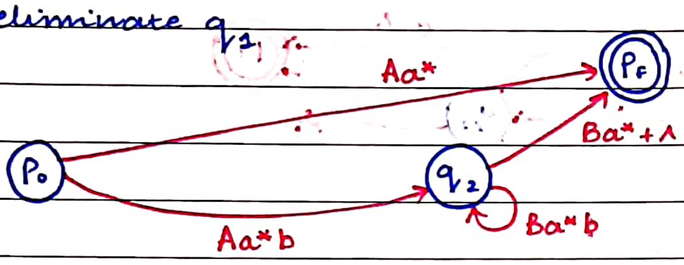
join every incoming with ~~an~~ every outgoing

2. eliminate q_3



Date:

3. eliminate q_1

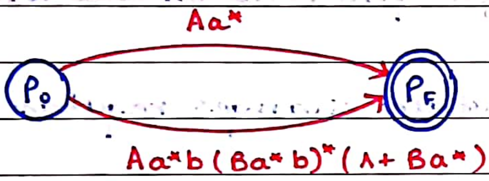


$a \cdot \Lambda = a$

$\Lambda + a \neq a$

$bb^* = b^+$

4. eliminate q_2



Expression: $(Aa^*b)(Ba^*b)^*(\Lambda + Ba^*) + Aa^*$

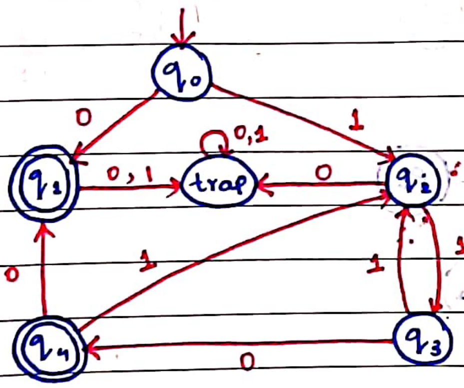
where $A = b^+a + a$

$B = (ab+b)b^*a + aa$

→ Week 4: Lecture 1

$(11 + 110)^*0$

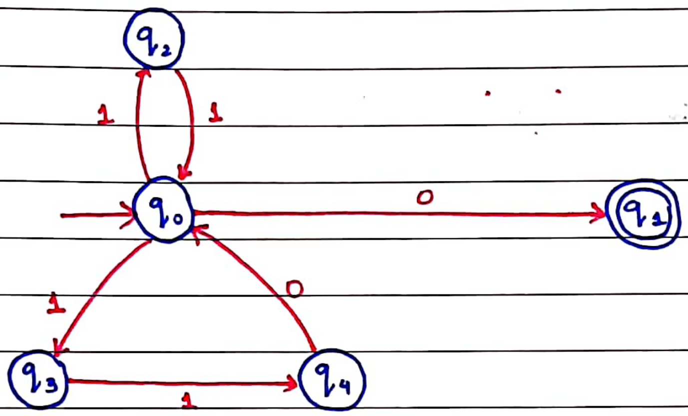
Form DFA



Date:

→ Non deterministic Finite Automata (NFA)

In DFA all input all states exactly 1 path, in NFA we can have 0 or multiple paths (max = # of states). If no path, it automatically crashe



We need not to worry about rejection
Implicit : Crash
Explicit : Stands on an intermediate state

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

for DFA,

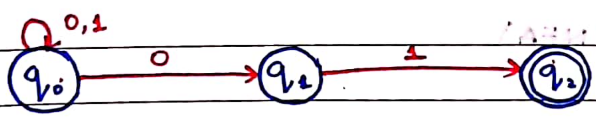
$Q \times \Sigma \rightarrow Q$

for NFA,

$Q \times \Sigma \rightarrow 2^Q$ power set

$\{q_0, q_1, q_2\} = \{ \phi,$
$\{q_0\}, \{q_1\}, \{q_2\},$
$\{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}$
$\{q_0, q_1, q_2\}$
$\}$

Question

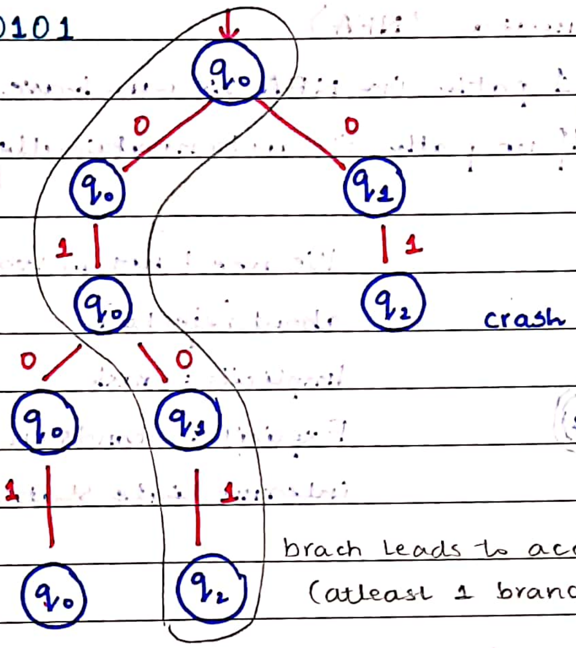


	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
q_2	ϕ	ϕ

Scanned with CamScanner

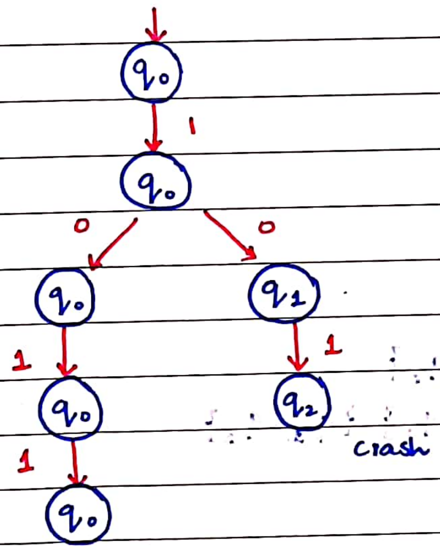
Date:

Check 0101 — Tree Traversal



brach leads to acceptance
(atleast 1 branch should lead to acceptance)

Check 1011



q_0 is intermediate state hence $1011 \notin L$

→ Extended Transition Function (NFA)

1. $\delta^*(q, \Lambda) \rightarrow q$
2. $\delta^*(q, ya) \rightarrow \bigcup\limits_{r \in \delta^*(q,y)} \delta(r, a)$

Every DFA is NFA but every NFA is not DFA

NFA 0 - n

DFA 1

Date:

Question: 01

$\delta^*(q_0, 01) = \bigcup \delta(r_1, 1) \rightarrow 1$ $\quad \rightarrow \delta(q_0, 1) \cup \delta(q_1, 1)$

$\{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

$r_1 \in \delta^*(q_0, 0) \quad \{q_0, q_1\}$

$\delta^*(q_0, \Lambda 0) = \bigcup \delta(r_2, 0) \rightarrow 2$ $\quad \delta(q_0, 0) \rightarrow \{q_0, q_1\}$

$r_2 \in \delta^*(q_0, \Lambda)$

$= \{q_0\}$

$\delta^*(q_0, \Lambda) \rightarrow \{q_0\}$ backtrack

$\{q_0, q_2\} \cap A \neq \phi$ hence 01 is part of the string

resultant final set is leaf node hence if they are in A, then accept.

Traverse 010

$\delta^*(q_0, 010) = \bigcup \delta(r_1, 0) \rightarrow (i)$ $\quad \delta(q_0, 0) \cup \delta(q_2, 0)$

$\{q_0, q_1\}$

$r_1 \in \delta^*(q_0, 01) = \{q_0, q_2\}$

$\delta^*(q_0, 01) = \bigcup \delta(r_2, 1) \rightarrow (ii)$ $\quad \delta(q_0, 1) \cup \delta(q_1, 1)$

$\{q_0, q_2\}$

$r_2 \in \delta^*(q_0, 0) \quad \{q_0, q_1\}$

$\delta^*(q_0, \Lambda 0) = \bigcup \delta(r_3, 0) \rightarrow (iii)$ $\quad \delta(q_0, 0) = \{q_0, q_1\}$

$r_3 \in \delta^*(q_0, \Lambda) = \{q_0\}$

backtrack

$\delta^*(q_0, \Lambda) \rightarrow q_0$

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

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

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

$= \{q_0, q_1\}$

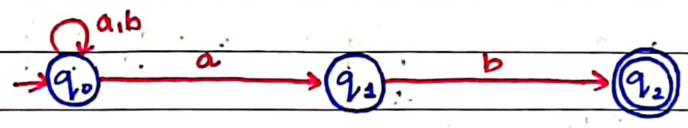
$010 \notin L$

Scanned with CamScanner

→ Week 4: Lecture 2

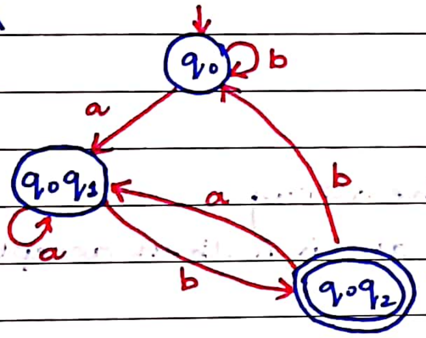
→ NFA to DFA

NFA



	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	q_2
q_2	ϕ	ϕ

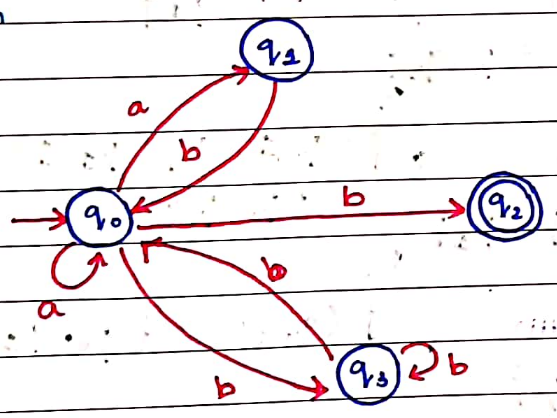
DFA



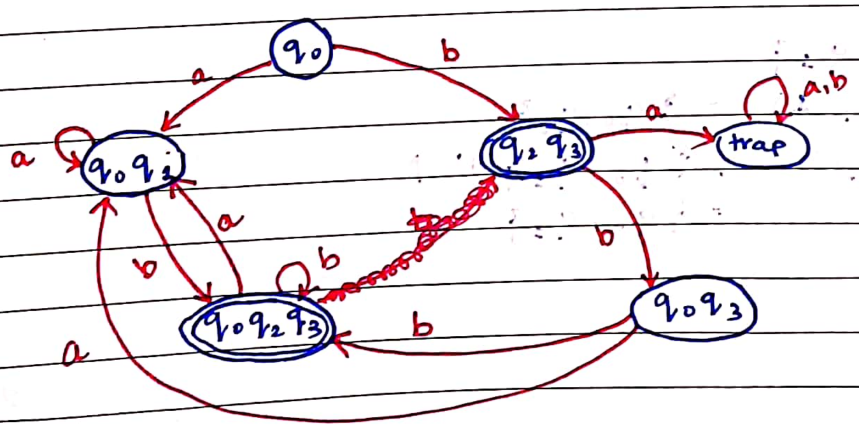
$$\delta(q_0q_2, b) = \delta(q_0, b) \cup \delta(q_2, b)$$
$$= \{q_0\}$$

Question

NFA



DFA

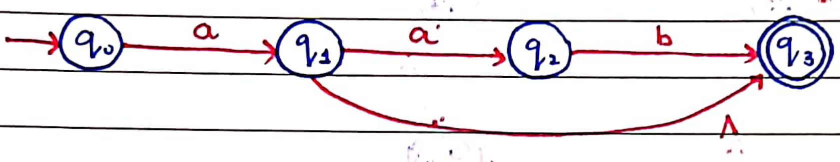


Date:

→ NFA -Λ

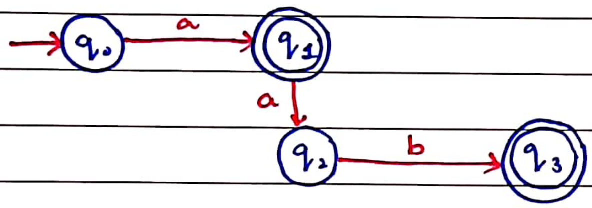
NFA + Λ transition

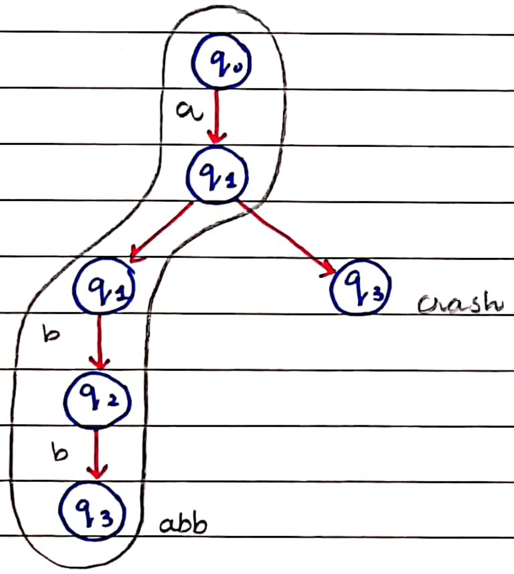
Example :

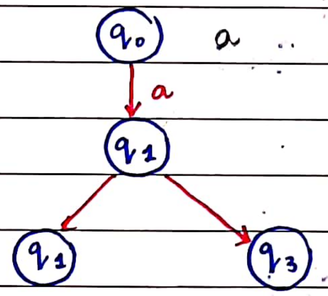


(a + aab)

is equivelant to



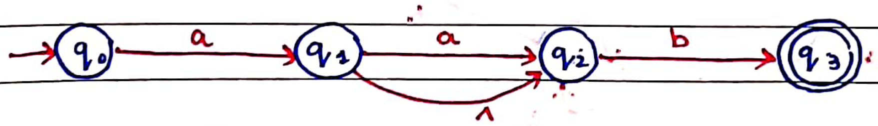




If null transition on a state, 2 options stay there or move forward, here we split

→ Λ- Closure

Without consuming input, how many states can be traversed.



$\Lambda\{q_0\} \rightarrow \{q_0\}$

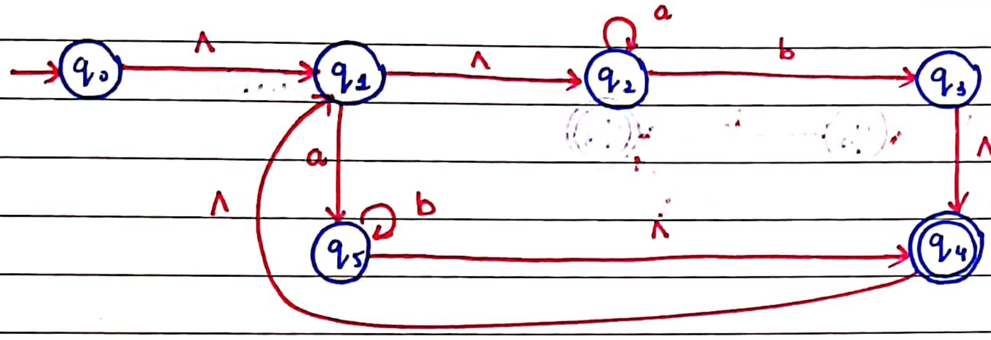
$\Lambda\{q_1\} \rightarrow \{q_1, q_2\}$

$\Lambda\{q_2\} \rightarrow \{q_2\}$

$\Lambda\{q_3\} \rightarrow \{q_3\}$

Date:

Null closure



$\Lambda\{q_0\} = \{q_0, q_1, q_2\}$

$\Lambda\{q_1\} = \{q_1, q_2\}$

$\Lambda\{q_2\} = \{q_2\}$

$\Lambda\{q_3\} = \{q_3, q_4, q_1, q_2\}$

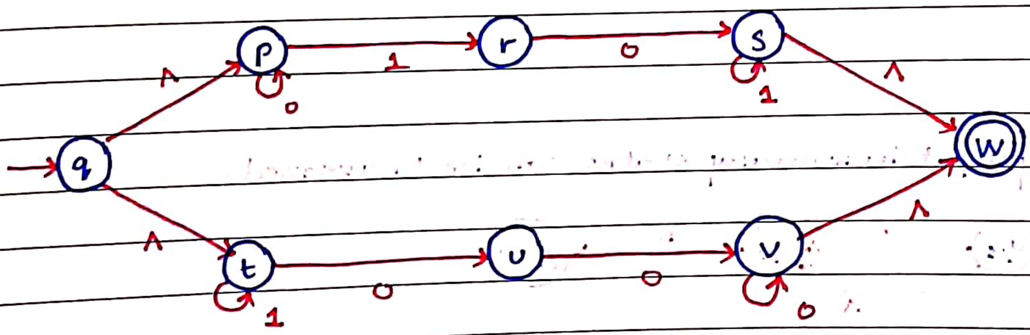
$\Lambda\{q_4\} = \{q_4, q_1, q_2\}$

$\Lambda\{q_5\} = \{q_5, q_4, q_1, q_2\}$

→ Extended Function

1. $\delta^*(q, \Lambda) \rightarrow \Lambda\{q\}$
2. $\delta^*(q, ya) = \Lambda\left\{ \bigcup_{r \in \delta^*(q,y)} \delta(r, a) \right\}$

Question



Date:

$$\delta^*(q,00) = \Lambda\{\bigcup_{r_1 \in \delta^*(q,0)} \delta(r_1,0)\} \rightarrow 1$$

$$\delta^*(q,10) = \Lambda\{\bigcup_{r_2 \in \delta^*(q,1)} \delta(r_2,0)\} \rightarrow 2$$

$$\delta^*(q,1) = \Lambda\{q\}$$
$$= \{q,p,t\}$$

Put in 2

$$\delta^*(q,0) = \Lambda\{\bigcup_{r_2 \in \{q,p,t\}} \delta(r_2,0)\}$$
$$= \Lambda\{\delta(q,0) \cup \delta(p,0) \cup \delta(t,0)\}$$
$$= \Lambda\{\phi \cup \{p\} \cup \{u\}\}$$
$$= \Lambda\{p,u\}$$
$$= \Lambda\{p\} \cup \Lambda\{u\}$$
$$= \{p,u\}$$

Put in 1

$$\delta^*(q,00) = \Lambda\{\bigcup_{r_1 \in \{p,u\}} \delta(r_1,0)\}$$
$$= \Lambda\{\delta(p,0) \cup \delta(u,0)\}$$
$$= \Lambda\{p,v\}$$
$$= \Lambda\{p\} \cup \Lambda\{v\}$$
$$= \{p,v,w\}$$

Does belong

Date:

→ 11

$\delta^*(q, 11) = \Lambda\{\bigcup_{r_1 \in \delta^*(q,1)} \delta(r_1, 1)\}$ → 1

$\delta^*(q, \Lambda 1) = \Lambda\{\bigcup_{r_2 \in \delta^*(q,\Lambda)} \delta(r_2, 1)\}$ → 2

$\delta^*(q, \Lambda) = \Lambda\{q\}$

$= \{q, p, t\}$

Put in 2

$\delta^*(q, 1) = \Lambda\{\bigcup_{r_2 \in \{q,p,t\}} \delta(r_2, 1)\}$

$= \Lambda\{\delta(q,1) \cup \delta(p,1) \cup \delta(t,1)\}$

$= \Lambda\{\phi \cup \{r\} \cup \{t\}\}$

$= \Lambda\{r\} \cup \Lambda\{t\}$

$= \{r, t\}$

Put in 1

$\delta^*(q, 11) = \Lambda\{\bigcup_{r_1 \in \{r,t\}} \delta(r_1, 1)\}$

$= \Lambda\{\delta(r,1) \cup \delta(t,1)\}$

$= \Lambda\{\Phi \cup t\}$

$= \Lambda\{t\}$

$= \{t\}$

Doesnot belong $11 \notin L$

Date:

⇒ Week 5: Lecture 1

→ Kleene's Theorem

Union

Concatenation

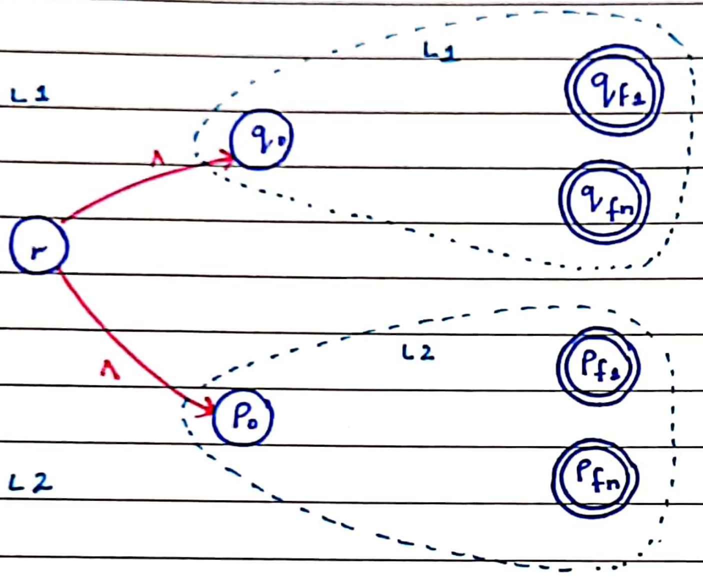
Kleene Star

L_1 L_2

$(Q_1, \Sigma, q_0, A_1, f_1)$ $(Q_2, \Sigma_2, p_0, A_2, f_2)$

$Q_1 \cap Q_2 = \phi$

→ Union



$L3 = L1 \cup L2$

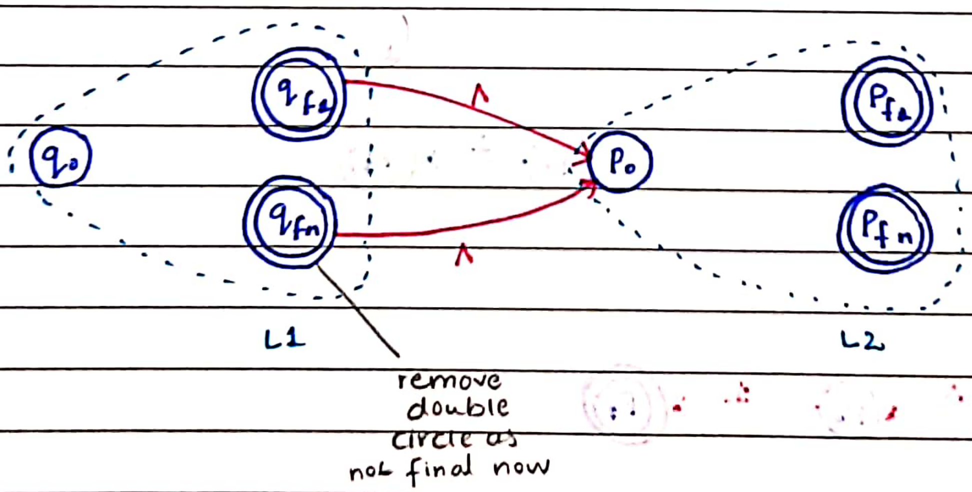
$(Q_3, \Sigma, r_3, A_3, f_3)$

$Q_3 = Q_1 \cup Q_2 \cup \{\Lambda\}$

$A3 = A1 \cup A2$

$f_3 = f_1 \cup f_2 \cup \{f_3(r, \Lambda) \rightarrow \{q_0, p_0\}$

→ Concatenation



Date:

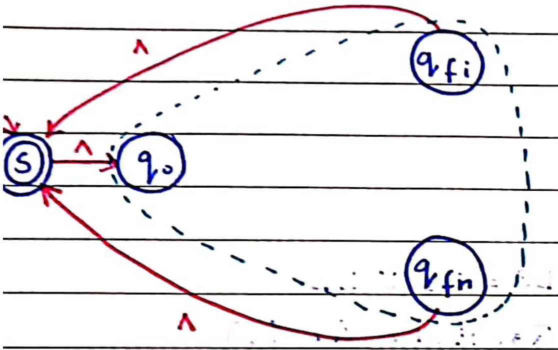
$L4 = L1 \cdot L2$

$(Q_4, \Sigma_1, q_0, A_4, \delta_4)$

$Q_4 = Q_1 \cup Q_2$

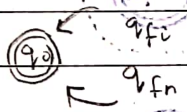
$\delta_4 = \delta_1 \cup \delta_2 \cup \{ \delta(q_{f_i}, \Lambda) \rightarrow \{p_0\} \}$ $1 \le i \le n$

→ Kleene Star



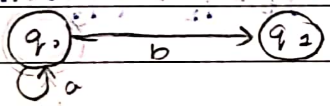
$L5 = L1^*$

cannot make q_0 final state

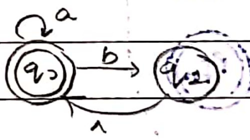


if loops present

eg



only a's not acceptable if

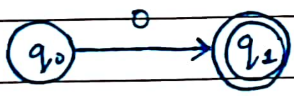


a's acceptable, language chang

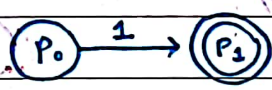
Every NFA is NFA null but every NFA null is not NFA

→ Question

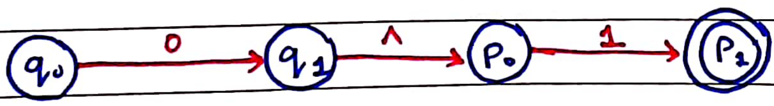
L1



L2



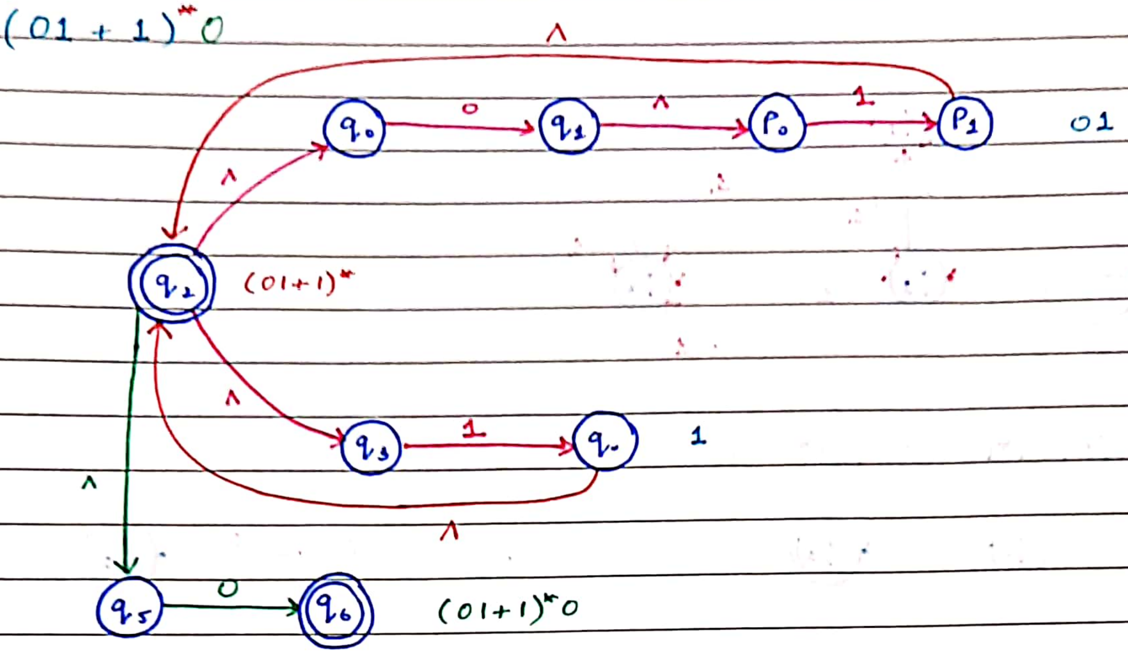
Concatenation (01)



HW. $(01+1^*1)^*0^*(10)^*$

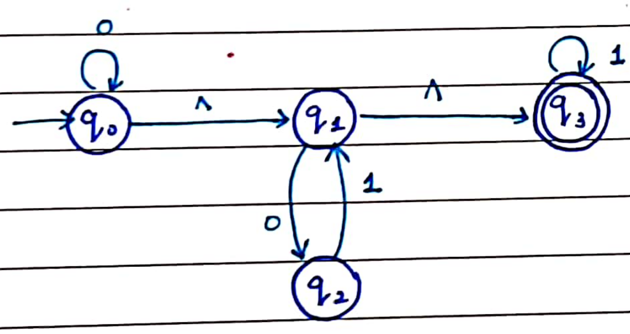
Date:

$(01+1)^*0$



→ Question

NFA null to NFA



$f^*(q_0,0) = \Lambda\{\cup f(r,0)\}$

$r \in f^*(q_0,\Lambda) = \{q_0,q_1,q_3$

$= \Lambda\{f(q_0,0) \cup f(q_1,0) \cup f(q_3,0)$

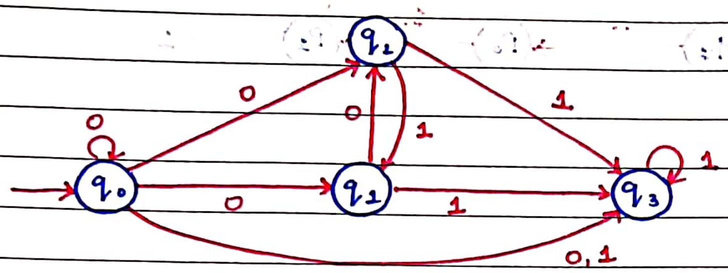
$= \Lambda\{q_0,q_2\} = \{q_0,q_1,q_2,q_3\}$

1. Fill transition table

	$f(q,0)$	$f(q,1)$	$\Lambda\{q\}$	$f^*(q,0)$	$f^*(q,1)$
q_0	q_0	ϕ	$\{q_0,q_1,q_3\}$	$\{q_0,q_1,q_2,q_3\}$	$\{q_3\}$
q_1	q_2	ϕ	$\{q_1,q_3\}$	$\{q_2\}$	$\{q_3\}$
q_2	ϕ	$\{q_1\}$	$\{q_2\}$	ϕ	$\{q_1,q_3\}$
q_3	ϕ	$\{q_3\}$	$\{q_3\}$	ϕ	$\{q_3\}$

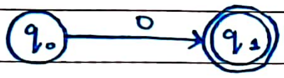
Date:

2 . Draw

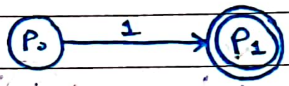


Question: $(01+1^*1)^* 0^* (10)^*$ HW

L1 =



L2



Scanned with CamScanner

Date:

→ Week 5 : Lecture 2

Distinguishable

(p, q) p = final q = intermediate

p = intermediate q = final

Indistinguishable

(p, q) p = q = intermediate

p = q = final

→ Equivalance Relation

A = set

1. Reflexive

$a \in A$ aRa

2. Symmetric

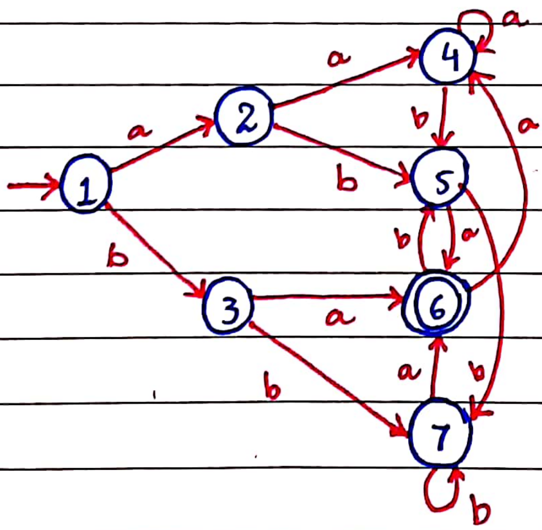
$a, b \in A$

aRb and bRa

3. Transitive

aRb and $bRc \Rightarrow aRc$

→ Minimization of DFA



Date:

	1	2	3	4	5	6	7
1	\	\	\	\	\	\	\
2		\	\	\	\	\	\
3	2	2	\	\	\	\	\
4			2	\	\	\	\
5	2	2		2	\	\	\
6	1	1	1	1	1	\	\
7	2	2		2		1	\

> 3R5 = 5R3
symmetric hence
ignore upper half
> ignore diagonal
because self relation

Step # 1: Mark the distinguishable relation
mark 1

Step # 2 :

if a,b one market 1 atleast, mark it 2

→ $f((2,1),a) = (4,2)$
$f((2,1),b) = (5,3)$
→ $f((3,1),a) = (6,2)$
→ $f((3,2),a) = (6,4)$
→ $f((4,1),a) = (4,2)$
$f((4,1),b) = (5,3)$
→ $f((4,2),a) = (4,4)$
$f((4,2),b) = (5,5)$
→ $f((4,3),a) = (4,6)$
→ $f((5,1),a) = (6,2)$
→ $f((5,2),a) = (6,4)$
→ $f((5,3),a) = (6,6)$
$f((5,3),b) = (7,7)$
→ $f((5,4),a) = (6,4)$

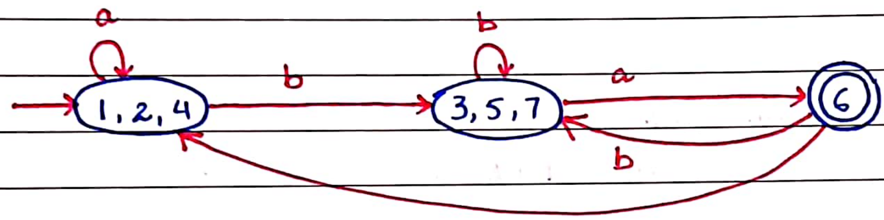
→ $f((7,1),a) = (6,2)$
→ $f((7,2),a) = (6,4)$
→ $f((7,3),a) = (6,6)$
$f((7,3),b) = (7,7)$
→ $f((7,4),a) = (6,4)$
→ $f((7,5),a) = (6,6)$
$f((7,5),b) = (7,7)$

Date:

Step # 3

- execute if any of the cell changed, i.e updation in Step # 2
- recheck all unmarked cells, if any updates do and execute Step 4 else stop here

merge column wise (empty cells + col no.)



→ Week 6: Lecture 2

Pumping Lemma

Regular

DFA → n states

if $x \in L$ and $|x| \geq n$

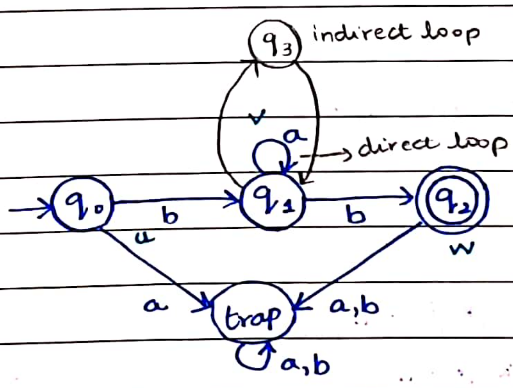
loop is present

$u = b$

$v = a$ → min len = 1

$w = b$

$x = uvw$
↓
loop



because $|v| \geq 1$, $u = \Lambda$
↑
$1 \leq |uv| \leq n$ → when $|uv| = n$, $w = null$

1. $|uv| \leq n$
2. $v \neq \Lambda$ ⇒ $|v| > 0$

Scanned with CamScanner

Date:

for all $i \geq 0$

$uv^iw \in L$

eg $u = b$ $v = a$ $w = b$

$uv^iw = ba^ib$ $v \neq \Lambda$, w, u can be Λ

$i = 0 \rightarrow bb$

$i = 1 \rightarrow bab$

$i = 3000 \rightarrow ba^{3000}b$

Question : $L = \{ a^m b^m$ where $m \geq 0$

Let L is regular then their must be DFA with n states

$|x| \geq n$

$x = a^n b^n$ take an instance w.r.t n

$|x| = n+n = 2n > n$

Devide x is 3 portions

$x = u.v.w$

$uv = a^n$

$w = b^n$

$v = a^k$ where $k > 0$ and $k \leq n$

$u = a^{n-k}$

$w = b^n$

$uv^iw \in L$

$uv^iw = (a^{n-k})(a^k)^i(b^n)$

$i = 1 \rightarrow a^{n-k}a^kb^n$

$= a^nb^n \in L$

$i = 2 \rightarrow (a^{n-k})(a^k)^2(b^n)$

$a^{n+k}b^n \notin L$

hence our assumption was wrong.

Date:

Question: $L = \{y \mid y \in \{0,1\}^* \text{ and } y \text{ is palindrome}\}$

$L \rightarrow DFA \rightarrow n$ states

$x = 0^n 0^n$

$|x| = 2n > n$

uvw

$uv = 0^n \qquad w = 0^n$

$v = 0^k$ where $k > 0$

$u = 0^{n-k}$

$uv^iw = (0^{n-k})(0^k)^i(0^n)$:

$i = 0 \rightarrow 0^{n-k}0^n$

In this instance violation can't occur for any value of i hence instance isn't correct

$x = 0^n 1 0^n$

1. $u = 0^{n-k}$

$v = 0^k$

$w = 10^n$

$uv = 0^n \rightarrow$ 2. $u = 0^{n-1}$ 3. $u = \Lambda$

$w = 10^n \qquad v = 0 \qquad v = 0^n$

$w = 10^n \qquad w = 10^n$

from first devision,

$uv^iw = (0^{n-k})(0^k)^i(10^n)$

$i = 0 \rightarrow 0^{n-k}10^n \notin L$

hence language not regular

Date:

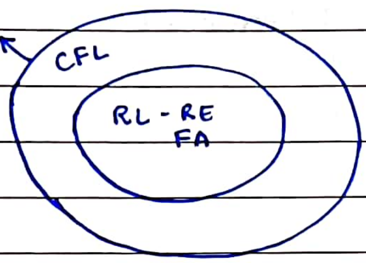
⇒ Week 7: Lecture 1

Context Free Languages (CFL) → CFG

Grammar ↓ base general

Context Free Grammar

PDA
Push Down Automata



< sentence >

<Noun Phrase> <NP> <Verb Phrase> <VP>

<article> <noun> <verb> <adverb>

eg The man is barking politely

↓

sentence follows the rule (although content doesn't make sense)
thus this is context free language.

- c++ is context sensitive

CFG : Context Free Grammar
(S, Σ, V, P)
4 tuple machine

V = Variable <verb><Noun> don't exist in Language but help

Σ = terminal / input alphabet set man, barking (part of language)

S = Starting Variable $S \in V$ (<sentence> above → root)

P = set of productions Rules

Date:

starting variable

<sentence> → <NP> <VP>
<NP> → <article> <Noun>
<VP> → <verb> <adverb>
<verb> → is talking / is barking
<noun> → man / dog
<article> → a / the
<adverb> → politely / loudly

} terminal

variables

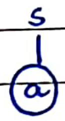
P: set of productions

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

→ Grammar has a treee
leaf nodes represent part of language
→ every string has diff tree

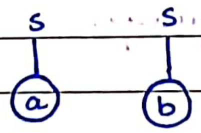
$L_1 = a$

$S \to a$



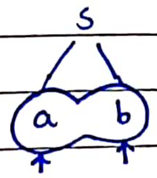
$L_2 = \{a, b\}$

$S \to a$
$S \to b$ } $S \to a/b$



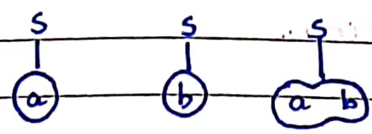
$L_3 = \{ab\}$

$S \to ab$



$L_4 = \{a, b, ab\}$

$S \to a / b / ab$



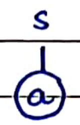
Date:

$L_3 = \{a, aa, aaa, aaaa, \dots\}$

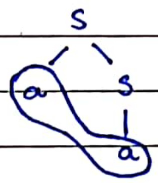
$S \to a$

$S \to aS$

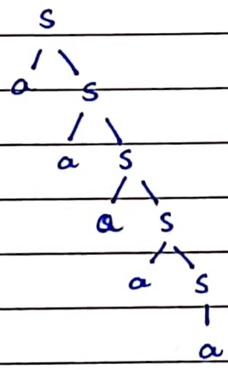
a



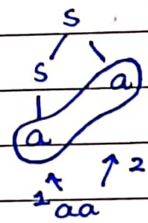
aa



aaaaa



aa

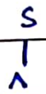


read left to right

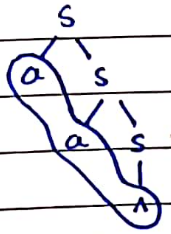
$L_4 = a^*$

$S \to \Lambda \mid aS$

Λ



aa



$aa\Lambda = aa$

$L_5 = (a+b)^*$

$S \to \Lambda / aS / bS$

abb

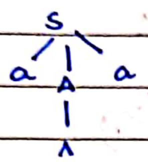
Date:

- $L_6 = \{x \mid x \in \Sigma^* \text{ and } x \text{ starts and ends with } a\}$

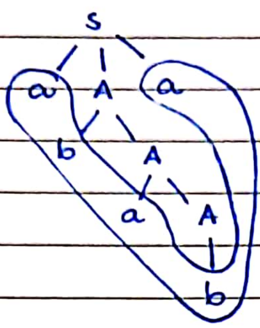
$S \rightarrow a \,/\, aAa$

$A \rightarrow \Lambda \,/\, aA \,/\, bA$

aa



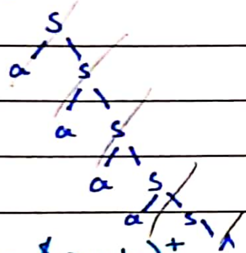
ababa

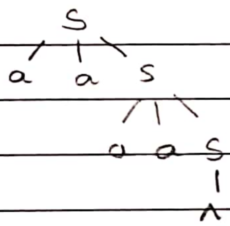


$L_7 = \{x \mid x \in a^* \text{ and } |x| \text{ is even}\}$

$S \rightarrow \Lambda \,/\, aaS$

aaaa

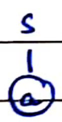




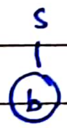
$L_8 = (a+b)^+$

$S \rightarrow a \,/\, b \,/\, aS \,/\, bS$

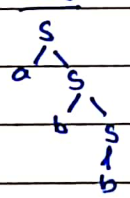
a



b



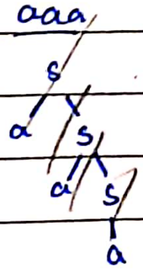
abb

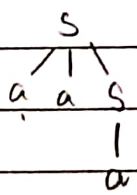


$L_9 = \{x \mid x \in a^* \text{ and } |x| \text{ is odd}\}$

$S \rightarrow a \,/\, aaS$

aaa





Date:

→ Week 7: Lecture 2

Develop CFG

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

- $L_1 = a^* + b^*$

  $A \to aA / \Lambda$

  $B \to bB / \Lambda$

  $S \to A / B$

- $L_2 = a^*bb$

  $A \to aA / \Lambda$

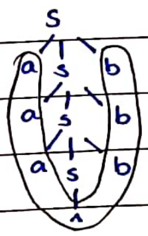
  $S \to Abb$

  $(S \to aS / bb)$

- $L_3 = a^i b^j$ where $i=j$

  $S \to aSb / \Lambda$

  eg a^3b^3



- $L_4 = a^i b^j$ where $i > j$

  $S \to aS / aaSb / a$

  $a^i b^j$

  $a^k \underbrace{a^i b^i}$ $a^i a^k b^i$

  $B \to aB / a$ $A \to aAb / \Lambda$ $S \to BA$

  $S \to aSb / A$

  $A \to aA / a$

- $L_5 = a^i b^j$ where $i > 2j$

  $a^k \underbrace{a^{2j} b^j}$ where $K > 0$

  $B \to aB / a$ $A \to aaAb / \Lambda$

  $S \to BA$

  $S \to aaSb / A$

  $A \to aA / a$

Date:

- $L_6 = a^i b^j$ where $i < j$

  $\underbrace{a^i b^i} b^j$

  $A \to aAb / \Lambda \qquad bB / \Lambda$ null acceptable (=)

  $S \to AB$

- $L_7 = a^i b^j c^k$ where $i = k$ and $j \geq 0$

  $A \to bA / \Lambda$

  $B \to aAc / A$

- L_8 = palindrome

  $S \to aSa / bSb / \Lambda / a / b$

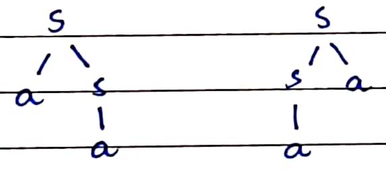
→ Week 8: Lecture 1

Ambiguous Language

multiple trees for same string

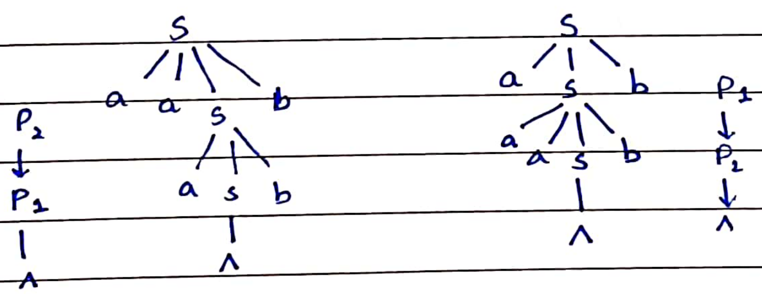
eg $S \to aS / Sa / a$

aa



1. $S \to aSb / aaSb / \Lambda$

   Ambiguous for aaabb



ambiguity because P1 + P2 are being used interchangeably. So allow P1 first then P2.

$S \to aSb / T$

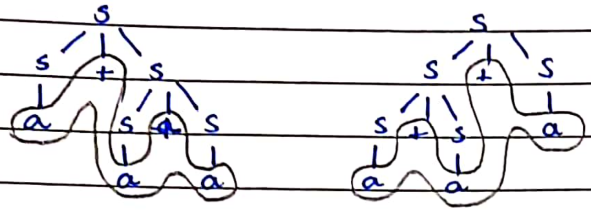
$T \to aaTb / \Lambda$

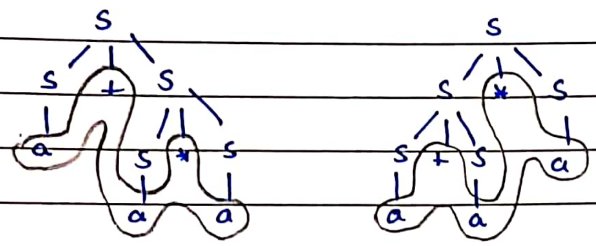
Date:

3. $S \rightarrow S+S \,/\, S-S \,/\, S*S \,/\, a$

$\Sigma = \{+, *, -, a\}$

a+a+a   a+a+a





a+(a*a)   (a+a)*a   mathematically correct

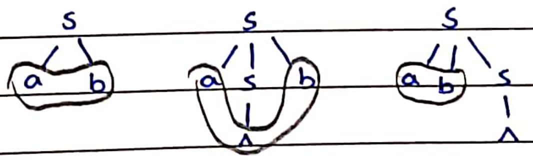
Ambiguity is due to expansion of S on both sides

$\therefore S \rightarrow S+T \,/\, S*T \,/\, S-T \,/\, a$

$T \rightarrow a$

4. $S \rightarrow aSb \,/\, abS \,/\, ab \,/\, \Lambda$

ab



Ambiguity is because of null helping to producing a string.

$\therefore S \rightarrow T \,/\, \Lambda$

$T \rightarrow aTb \,/\, abT \,/\, ab$

Date:

$L_1 \downarrow G_1 \downarrow (S_1, \Sigma_1, V_1, P_1)$ $\qquad$ $L_2 \downarrow G_2 \downarrow (S_2, \Sigma_2, V_2, P_2)$

$V_1 \cap V_2 = \phi$

- $L_3 = L_1 \cup L_2$

$\downarrow$

G_3

$\downarrow$

(S_3, Σ, V_3, P_3)

$S_3 \rightarrow S_1 / S_2$

$V_3 = V_1 \cup V_2 \cup S_3$

$P_3 = P_1 \cup P_2 \cup \{ S_3 \rightarrow S_1 / S_2 \}$

- $L_4 = L_1 L_2$

$\downarrow$

G_4

$\downarrow$

(S_4, Σ, V_4, P_4)

$S_4 \rightarrow S_1 S_2$

$V_4 = V_1 \cup V_2 \cup \{S_4\}$

$P_4 = P_1 \cup P_2 \cup \{ S_4 \rightarrow S_1 S_2 \}$

- $L_5 = L_1^*$

$\downarrow$

G_5

$\downarrow$

(S_5, Σ, V_5, P_5)

$S_5 \rightarrow S_1 S_5 / \Lambda$

$V_5 = V_1 \cup \{S_5\}$

$P_5 = P_1 \cup \{ S_5 \rightarrow S_1 S_5 / \Lambda \}$

Date:

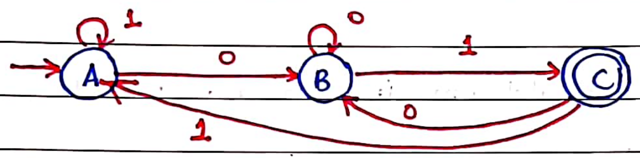
$(10+11)^* 01$

$S_1 \to 10$ $A \to S_1 / S_2$

$S_2 \to 11$ $B \to AB / \Lambda$

$S_3 \to 01$ $S \to BS_3$

$S_4 \to S_3 S_4 / \Lambda$



ends with ab.

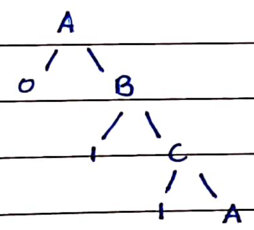
input   dest state

$A \to 1A / 0B$

$B \to 0B / 1C$

$C \to 0B / 1A$

011



$C \to \Lambda$ some consider Λ as terminal

OR

$B \to 1$ ✓ (chose this for RG)

Regular Grammar

Variable → (terminal)(Variable)

Variable → terminal

Date:

→ Week 8: Lecture 2

Chomsky Normal Form (CNF)

(1) Remove useless production

S
|
S
|
S
|
S
|
a

$S \to A/B$

$A \to aA$ → useless because no termination / infinite loop
recursive call without basecase

$B \to bB/b$

(2) Remove Λ production

if null acceptable language, Λ production should come in starting variable
else Λ shouldnot be in any production.

$S \to BAB$ ⟶ $S \to BAB / BB$
$A \to aA / \Lambda$ ⟶ $A \to aA / a$
$B \to bB / b$ ⟶ $B \to bB / b$

this language doesn't accept Λ

Example

1. $S \to BAB$ ⟶ $S \to BAB / AB / BA / A$
$A \to aA / a$ ⟶ $A \to aA / a$
$B \to bB / \Lambda$ ⟶ $B \to bB / b$

2. $S \to BAB$ ⟶ $S \to BAB / AB / BA / A$
$A \to aA / Bc / a$ ⟶ $A \to aA / Bc / a / c$
$B \to bB / \Lambda$ ⟶ $B \to bB / b$

Λ can't be removed from S

3. $S \to BA$ ⇒ $S \to BA$ ⟶ $S \to BA / B / A / \Lambda$
$A \to aA / BB / a$ ⇒ $A \to aA / BB / a / B / \Lambda$ ⟶ $A \to aA / BB / a / B$
$B \to bBc / \Lambda$ ⇒ $B \to bBc / bc$ ⟶ $B \to bBc / bc$

Date:

③ Remove Unit Production

A single Variable is producing a single variable

eg. S → B / CB   S → CB / bB / b
B → bB / b   ⇒   B → bB / b
C → aC / a   C → aC / a

eg. Apply first 3 steps on grammar

S → AACD
A → aAb / Λ
C → aC / C / a
D → aDa / bDb / Λ

Step 1 : S → AACD
A → aAb / Λ
C → aC / a
D → aDa / bDb / Λ

Step 2 : S → AACD / ACD / CD / AAC / AC / C
A → aAb / ab
C → aC / a
D → aDa / bDb / aa / bb

Step 3 : S → AACD / ACD / CD / AAC / AC / aC / a
A → aAb / ab
C → aC / a
D → aDa / bDb / aa / bb

Date:

④ Convert right side to string of variables or a single terminal

$S \to AACD \mid ACD \mid CD \mid AAC \mid AC \mid aC \mid a$
$A \to aAb \mid ab$
$C \to aC \mid a$
$D \to aDa \mid bDb \mid aa \mid bb$

$X_a \to a$
$X_b \to b$

$S \to AACD \mid ACD \mid CD \mid AAC \mid X_aC \mid a \mid AC$
$A \to X_aAX_b \mid X_aX_b$
$D \to X_aDX_a \mid X_bDX_b \mid X_aX_a \mid X_bX_b$
$C \to X_aC \mid a$

⑤ Convert right side to exactly 2 variables or single terminal

$S \to AACD$		
$S \to AT_1$	$S \to AT_2$	$S \to X_aC$
$T_1 \to AT_2$	$S \to CD$	$S \to a$
$T_2 \to CD$	$S \to AT_3$	$S \to AC$
	$T_3 \to AC$	

$A \to X_aT_4$	$D \to X_aT_5$	$D \to X_aX_a$
$T_4 \to AX_b$	$T_5 \to DX_a$	$D \to X_bX_b$
$A \to X_aX_b$	$D \to X_bT_6$	$C \to X_aC$
	$T_6 \to DX_b$	$C \to a$

Date:

→ Week 9 : Lecture 1

Alphabet

↓

Words

↓

Language

1. FA ← Regular language
   - ↳ DFA
   - ↳ NFA
   - ↳ NFA Λ
2. RE

Non Regular language

Regular language → CFL ← Non Regular language

CFL → 1. CFG
2. Push Down Automata (PDA)
   - ↳ DPDA
   - ↳ NPDA (more powerful)

↓

FA + auxiliary memory (Stack)
- ↳ FILO
- ↳ LIFO

Traversal

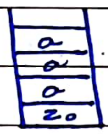
$L = \{a^n b^n \text{ where } n > 0\}$

eg aaabbb

push a's in stack + consume input

consume b's and pop b from stack

if Z_0 in end language acceptable



• Push

input ← $a, \overset{\uparrow \text{TOS}}{Z_0} / \overbrace{aZ_0}^{\text{top 2 elements of stack}}$

$a, a / aa$

Date:

Pop

input b/a $\xrightarrow{}$ Λ ($\rightarrow$ pop)

(a = TOS)

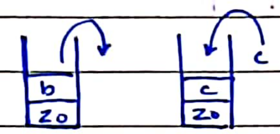
One input and 2 actions

eg a, b / C   if a comes, tos b, replace b with c

two transitions for this

a, b / Λ consume a, tos b, pop (Λ) it

Λ, x / cx   without consuming any input, no matter what tos, put c on top (represented by x as we don't know what's under b)



a, b / Λ Λ, x / cx

here $x = z_0$

PDA

$Q \times \Sigma \rightarrow Q$

PDA = FA + Stack

PDA → 7-tuple

FA → 5 → ($Q, \Sigma, q_0, A, \delta$)

Stack → 2 → (T, z_0)

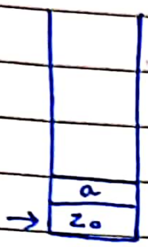
z_0 → stack element

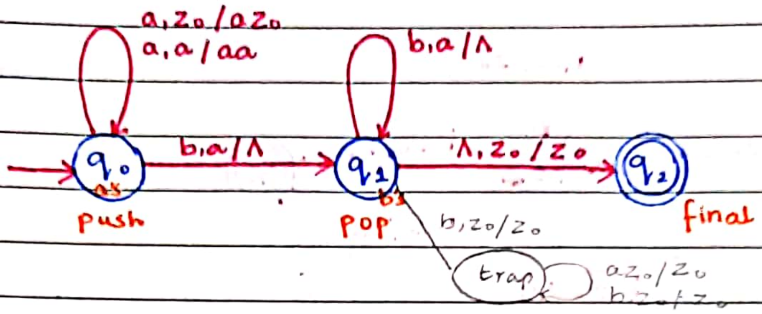
$(Q, \Sigma, q_0, A, T, z_0, \delta)$

$Q \times \{\Sigma \cup \Lambda\} \times \{T\} \longrightarrow Q \times T^*$

Date:

$L = \{ a^n b^n \text{ where } n > 0 \}$





If not mentioned in question explicitly only represent accepting states, ignore rejections

Traversal

current state ↑ string ↑ stack ↑

$(q_0, aabb, z_0)$

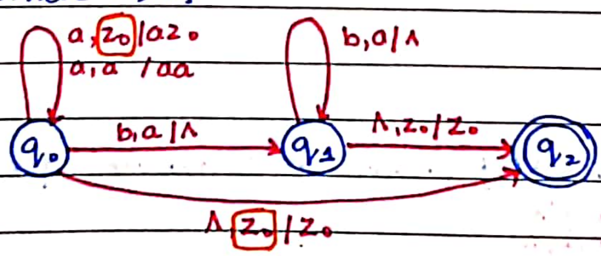
↓

push	$(q_0, abb, a z_0)$
push	$(q_0, bb, aa z_0)$
pop	$(q_1, b, a z_0)$
pop	(q_1, Λ, z_0)
final	(q_2, Λ, z_0)

input consumption is necessary however stack may/maynot be empty.

for deterministic if state is same, either input or top of stack has to be different. Λ cannot be with any else ie whatever the input

eg $L = \{ a^n b^n \text{ where } n \geq 0 \}$

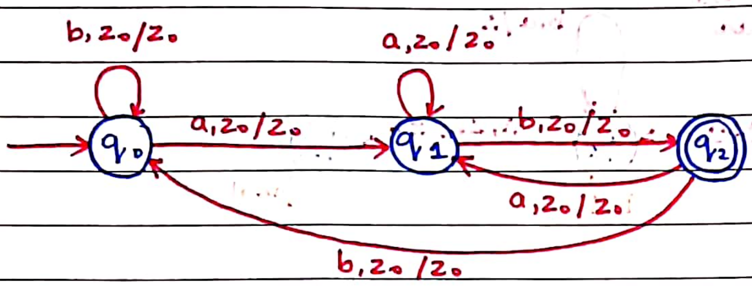


Due to these deterministic PDA cannot be developed. We have 2 options at q_0 for input 'a' and top of stack 'z_0'

Date:

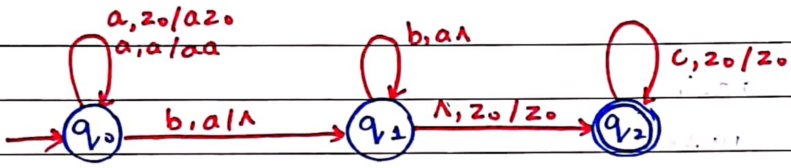
→ Question:

L_1 = { ending with ab }



done without stack

$L_2 = \{a^n b^n c^m$ where $n > 0$ and $m \geq 0\}$



Traversal

(q_0, abc, z_0)

(q_0, bc, az_0)

(q_1, c, z_0)

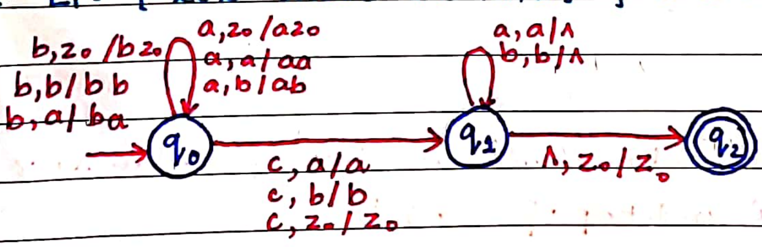
(q_2, c, z_0)

(q_2, Λ, z_0)

→ Week 9, Lecture 1

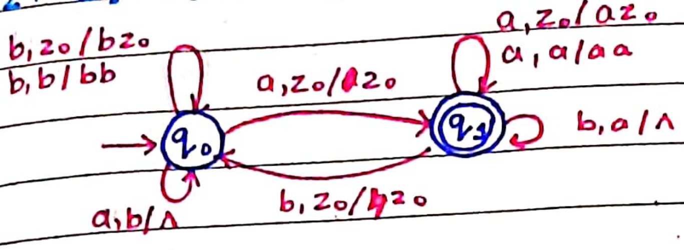
Develop PDA for

1. L_1 = { xcx^r where $x \in \{a, b\}^*$ } where x^r is reverse of x



Date:

2. $L_2 = \{ x \mid n_a(x) > n_b(x)$ where n_a is number of a's in $x \}$



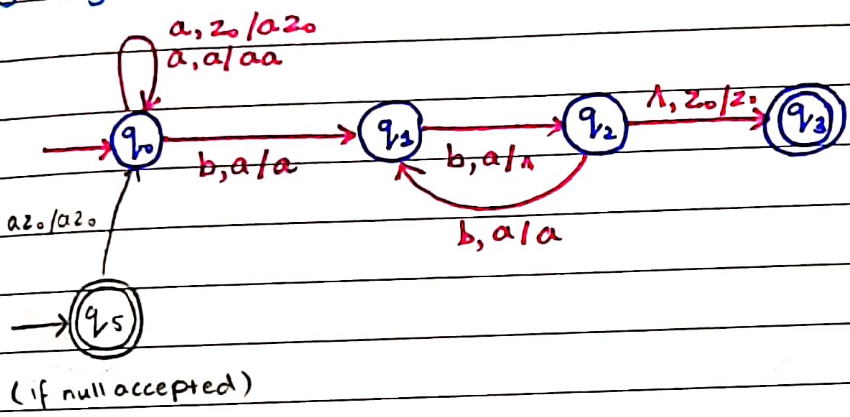
$q_0 : n_b \geq n_a$

$q_1 : n_a > n_b$

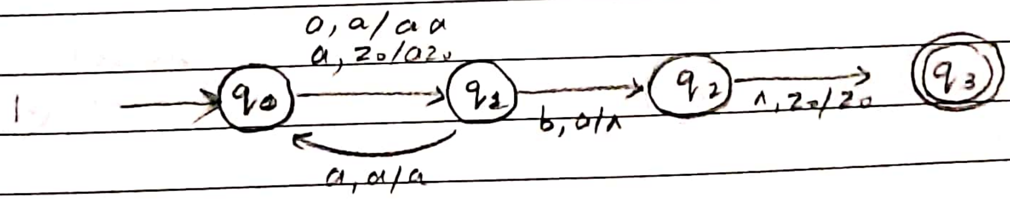
3. $L_3 = a^i b^{2i}$

push 2 a's for each a

a, z₀ / aaz₀
↓
1. a, z₀ / az₀
2. Λ, a / aa



4. $L_4 = a^{2i} b^i$



* If question says develop PDA, your choice either detministic or non deterministic.

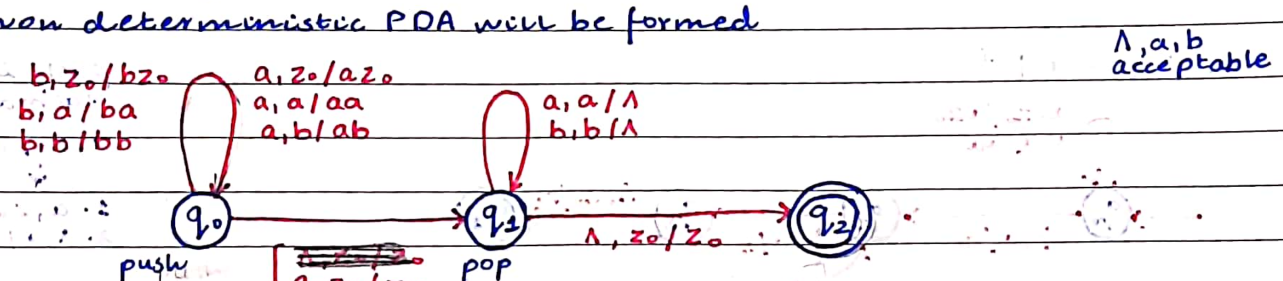
ab ba

Date:

→ Week 10 : Lecture 1

Develop PDA for palindrome $\Sigma = \{a, b\}$

- We don't know what the middle element is hence deterministic PDA can't be developed.
- We have to take partial decession if element is in first half, second half or is middle.
- If odd, simply consume middle element else check.
- A non deterministic PDA will be formed



single element { a, Z_0 / Z_0 ; b, Z_0 / Z_0

odd length { a, a / a ; a, b / b ; b, a / a ; b, b / b

this also

even length { Λ, a / a ; Λ, b / b   Λ, Z_0 / Z_0

Traversal

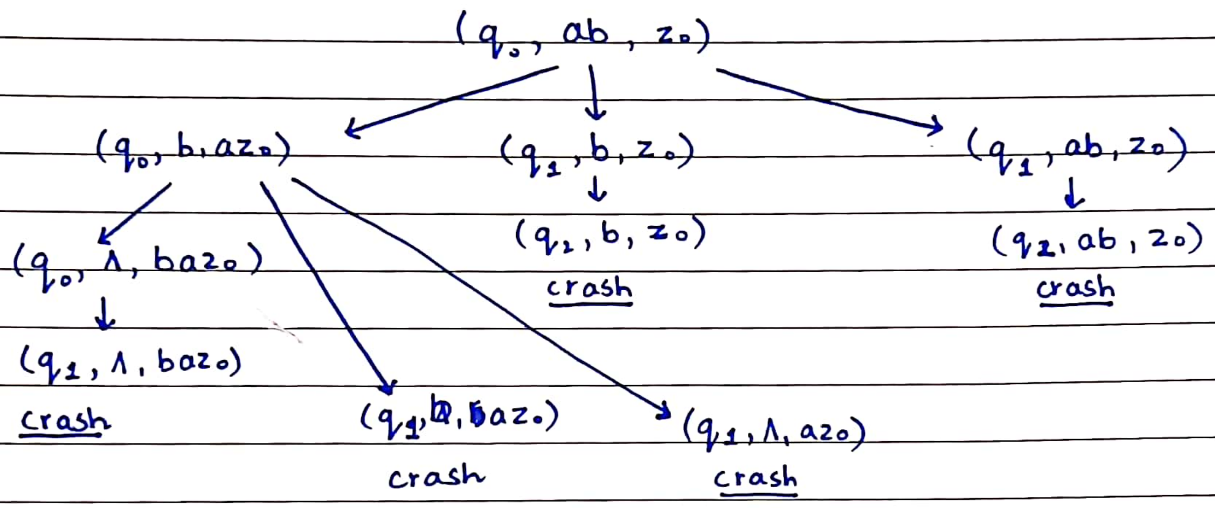
(q_0, abba, Z_0)
- (q_0, bba, aZ_0)
  - (q_0, ba, baZ_0)
    - (q_0, a, bbaZ_0)
    - (q_1, a, baZ_0) crash
    - (q_1, ba, baZ_0)
      - (q_1, a, aZ_0)
      - (q_1, Λ, Z_0)
      - (q_2, Λ, Z_0) Accept
  - (q_1, ba, aZ_0) crash
  - (q_1, bba, aZ_0) crash
- (q_1, bba, Z_0)
  - (q_2, bba, Z_0) crash
- (q_1, abba, Z_0)
  - (q_2, abba, Z_0) crash

Scanned with CamScanner

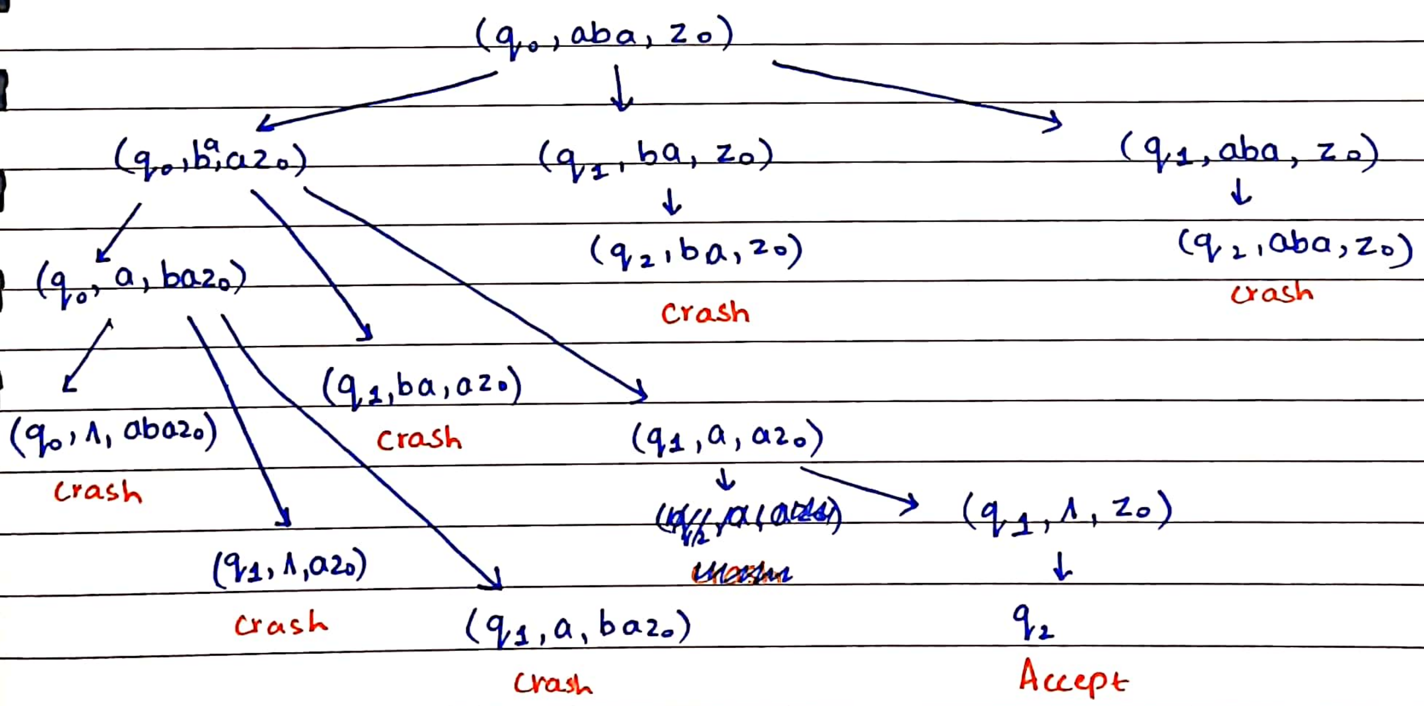
Date:

- If branch leads to acceptance, ignore the rest
  however for rejection show all branches crash or explicit rejection

Traversal - ab



Traversal - aba



Date:

⇒ Week 10: Lecture 2

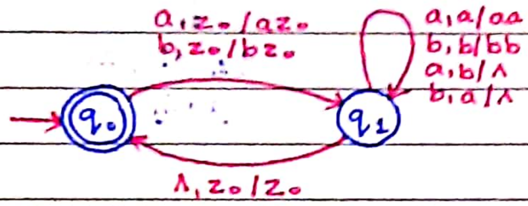
Develop PDA and CFG

$L_1 = \{x \mid n_a(x) = n_b(x)\}$

$L_2 = \{x \mid x = a^i b^j c^j d^i \text{ where } i,j \geq 0\}$

L_1:

$S \rightarrow aSb / bSa / \Lambda$

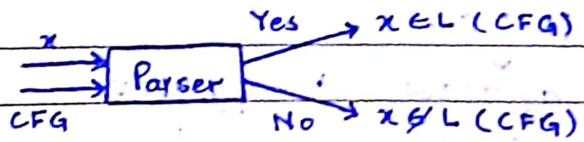


L_2:

$S \rightarrow aBd / aSd$

$B \rightarrow bBc / \Lambda$

⇒ Parser



Date:

Top Down Parser

1. Push the starting variable S in stack
   Repeat till input and stack are empty
2. If the TOS element is a variable
   ↳ replace non deterministically the TOS element with its right ~~most~~ side
   else if the TOS element is terminal, compare with current input
   ↳ if same pop
   ↳ else end crash

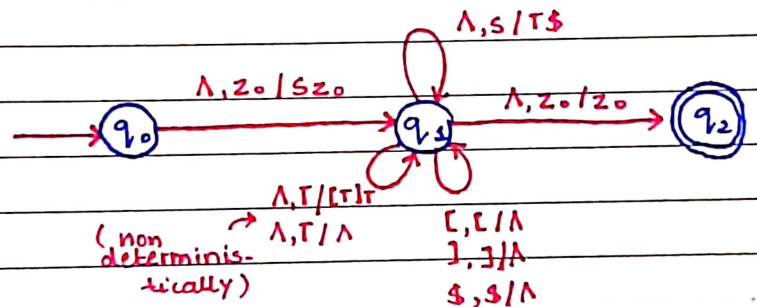
TD parser

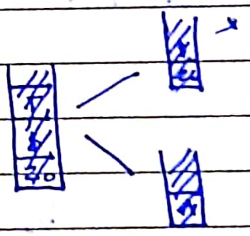
$S \to T\$$

$T \to [T]T \mid \Lambda$

Terminals : { [,], $ }

Variable : { S, T }





$(q_0, []\$, Z_0)$

↓

$(q_1, []\$, SZ_0)$

↓

$(q_1, []\$, T\$Z_0)$

$T \to \Lambda$ ↙ ↘ $T \to [T]T$

$(q_1, []\$, \Lambda\$Z_0)$ crash

$(q_1, []\$, [T]T\$Z_0)$

$(q_1,]\$, T]T\$Z_0)$

$T \to \Lambda$ ↓

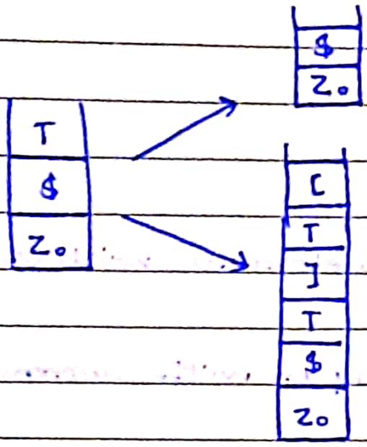
$(q_1,]\$, \Lambda]T\$Z_0)$

$(q_1, \$, T\$Z_0)$

$(q_1, \$, \$Z_0)$

(q_2, Λ, Z_0) accept

Date:



→ Week 11 : Lecture 1

Bottom up Parser
(grammar should be in chomsy normal form)

S → XY
X → XA / a / b
Y → AY / a
A → a

Parse the string baaa

(fill diagonally)

b	a	a	a
			XYA 4,1
		XYA 3,1	SXY 3,2
	XYA 2,1	SXY 2,2	SXY 2,3
X 1,1	SX 1,2	SX 1,3	SX 1,4

→ S here so accept

- For (1,2) we take cartesian product of (1,1) and (2,1)
  {X} x {X, Y, A}
  {XX, XY, XA} = {S, X}
- For (1,3) we take cartesian product of (1,1)(2,2) ∪ (1,2)(3,1) (left most and immediate top then next element so on)

Date:

If the root node exists in the last cell i.e (1,4), we say that the string is accepted by the Language.

Parse the string babaa

b	a	b	a	a
				X,Y,A (5,1)
			X,Y,A (4,1)	S,X,Y (4,2)
		X (3,1)	S,X (3,2)	S,X (3,3)
	X,Y,A (2,1)	Φ (2,2)	Φ (2,3)	Φ (2,4)
X (1,1)	S,X (1,2)	Φ (1,3)	Φ (1,4)	Φ (1,5)

Therefore the string babaa is not acceptable

$\Lambda . X = X$

$\Phi . X = \Phi$

$(1,2) = \{X\} \times \{X, Y, A\}$

$= \{XX, \underset{\downarrow}{XY}, \underset{\downarrow}{XA}\}$
$\quad\ X \qquad S \qquad X$ $= \{S, X\}$

$(2,2) = \{X, Y, A\} \times \{X\}$

$= \{XX, YX, AX\} = \Phi$

$(3,2) = \{X\} \times \{X, Y, A\}$

$= \{XX, XY, XA\} = \{S, X\}$

$(4,2) = \{X, Y, A\} \times \{X\ Y\ A\}$

$= \{XX, \underset{\downarrow}{XY}, \underset{\downarrow}{XA}, YA, YX, YY, AX, \underset{\downarrow}{AY}, AA\}$
$= \qquad S \qquad X \qquad\qquad Y$ $= \{S, X, Y\}$

$(1,3) = (1,1) \times (2,2) \cup (1,2) \times (3,1)$

$\Phi \cup \Phi = \Phi$

Scanned with CamScanner

Date:

→ Week 11: Lecture 2

Develop PDA

$L_1 : \{ a^j b^j c^j$ where $j \geq 0 \}$

Non context free language as comparison of more than 2 characters can't be done through 1 stack. Done through Tuning machine. So PDA doesnt exist.

$L_2 : \{ SS$ where $S \in \{a,b\}^* \}$

Non-CFL

- Height of a tree $= h$
- If p variables, then maximum height without repetition $h = P+1$
- No. of Leaf nodes (max) 2^{h-1} (2 for binary)

  generally a^{h-1}

If binary without repetition $h = P+1$ (repetition in a single branch)

max no. of leaf nodes $2^{h-1} = 2^{P+1-1} = 2^P$

Nodes (leaf) represent terminals

$2^P \rightarrow$ strings generated without repetition

if $x \in L$,

$|x| \geq 2^{P+1} \rightarrow n$

repetition occurs

$h = P+2$

$2^{P+2-1} = 2^{P+1}$

n is the min no. of leaf nodes with recursion

Pumping Lemma 1 is used to prove if a language is regular or not.

Pumping Lemma 2 is used to prove if a language is context free or not.

Date:

Pumping Lemma 2

$u \in L$

$|u| \geq n$

recursion

$u = v\underline{w}xyz$

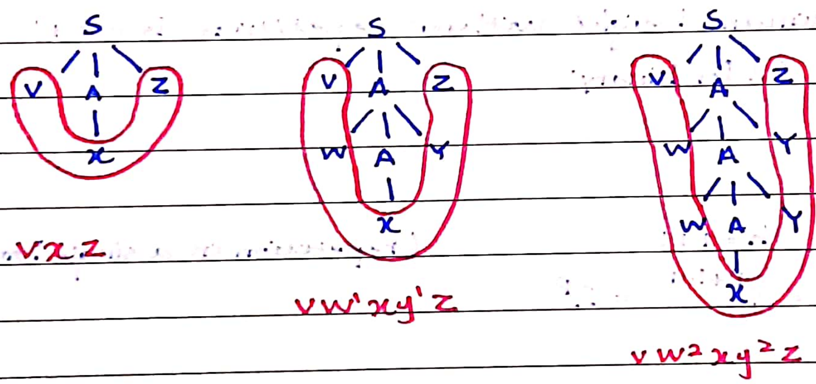
1. $|wy| > 0$ (w,y can't be null simultaneously else no recursion so not part of language)
2. $|wxy| \leq n$

$S \to vAz$

$A \to wAy / x$

where S and A are variables

v, z, w, y, x are terminals



Question

$L = \{a^j b^j c^j \text{ where } j \geq 0\}$

Let L in CFL $\to$ CFG with P variables

$n = a^{P+1}$

$u = a^n b^n c^n$

$|u| = n + n + n = 3n > n$

Date:

Devisions

$v = \Lambda$ $\quad$ $wxy = a^n$ $\quad$ $z = b^n c^n$

$v = a^n$ $\quad$ $wxy = b^n$ $\quad$ $z = c^n$

$v = a^n b^n$ $\quad$ $wxy = c^n$ $\quad$ $z = \Lambda$

$v = a^n b^{n/2}$ $\quad$ $wxy = b^{n/2}$ $\quad$ $z = c^n$

Can take any division from which a conclusion can be made

$v = a^n$, $wxy = b^n$, $z = c^n$

$w = b^{n/2}$, $x = \Lambda$, $y = b^{n/2}$

$vw^ixy^iz = (a^n)(b^{n/2})^i(\Lambda)(b^{n/2})^i c^n$

$= a^n (b^{n/2})^0 (\Lambda)(b^{n/2})^0 c^n \notin L$ $\quad$ $(i=0)$

L is non context free Language

→ Week 12 : Lecture 1

Prove using pumping Lemma 2

$L = \{ ss \text{ where } s \in \{a,b\}^* \}$

Let L is CFL ⇒ CFG with P variables

$u = a^n a^n$

$|u| = n+n = 2n > n$

• $v = \Lambda$

$wxy = a^n$ $\quad$ $w = a^{n/2}$, $x = \Lambda$, $y = a^{n/2}$

$z = a^n$

$vw^ixy^iz = (a^{n/2})^i (a^{n/2})^i a^n$

$=$

• $v = a^n$, $wxy = a^n$, $z = \Lambda$

$w = a^{n-k}$, $x = a^k$, $y = \Lambda$

$vw^ixy^iz = (a^n)(a^{n-k})^i(a^k)(\Lambda)^i(\Lambda)$

$i=0$ $\quad = a^{n+k}$ $\quad$ (if k even acceptable can't conclude)

if k odd can conclude

Date:

- $v = a^n$, $w = a^{n/2}$, $x = \Lambda$, $y = a^{n/2}$, $z = \Lambda$

$vw^i xy^i z = (a^n)(a^{n/2})^i \Lambda (a^{n/2})^i (\Lambda)$

$i = 2$ $(a^n) a^n a^n$

$a^{n+\frac{n}{2}} \; a^{n+n/2}$

$v = a^n$	$v = \Lambda$	$v = \Lambda$
$wxy = a^n$	$wxy = a^n$	$wxy = a^{n/2}$
$z = \Lambda$	$z = a^n$	$z = a^{n/2} a^n$

↓

$w = \Lambda$	$w = a^{n/2}$	$w = a^{n-k}$	$w = a^{n-k}$
$x = a^{n/2}$	$x = \Lambda$	$x = a^k$	$x = \Lambda$
$y = a^{n/2}$	$y = a^{n/2}$	$y = \Lambda$	$y = a^k$

for all ~~instances~~ devisions we should be able to conclude. If no conclusion, instance is incorrect.

$S = a^n b^n$

$S.S = a^n b^n a^n b^n$

$|SS| = n+n+n+n = 4n > n$

$v = a^n$ $wxy = b^n$ $z = a^n b^n$ $w = b^{n/2}$ $x = \Lambda$ $y = b^{n/2}$

$vw^i xy^i z = (a^n)(b^{n/2})^i \Lambda (b^{n/2})^i a^n b^n$

$i = 2 \Rightarrow a^n b^n b^n a^n b^n \notin L$

Disapproved (In this every devision will lead to unequal a's b's so not part of language)

Assumption was wrong, L is not CFL.

Date:

If L_1 and L_2 are CFL

1. $L_1 \cup L_2$ CFL, closed under union $S \rightarrow S_1 / S_2$
2. $L_1 L_2$ CFL $S \rightarrow S_1 S_2$
3. L_1^* CFL $S \rightarrow SS_1 / \Lambda$
4. L_1^r (reverse of L_1) CFL $a^n b^n \rightarrow b^n a^n$
5. $L_1 \cap L_2$ non CFL $L_1 = \{a^n b^n c^k, n,k > 0\}$ $L_2 = \{a^k b^n c^n, n,k > 0\}$
6. $\bar{L_1}$ non CFL $L_1 \cap L_2 = \{a^n b^n c^n \ \ n > 0\}$ non CFL
7. $L_1 - L_2$ non CFL

for $\bar{L_1}$

Let CFL is closed under complement

$$A \cap B = \underbrace{\overline{\underset{CFL}{\bar{A}} \cup \underset{CFL}{\bar{B}}}}_{CFL}$$

but not CFL as intersection

If L_1 is CFL and L_2 regular

1. $L_1 \cup L_2$ CFL
2. $L_1 . L_2$ CFL
3. L_1^* CFL
4. $L_1 \cap L_2$ RL CFL
5. $L_1 - L_2$ CFL

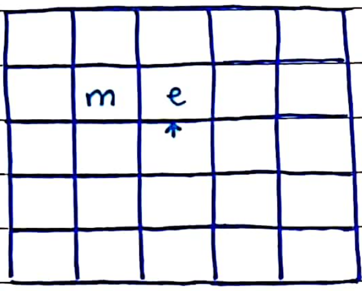
Date:

Week 13: Lecture 1

Turing Machine

(non CFL)

Σ = {#



focus point:

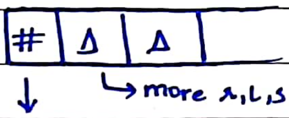
read, write, replace

it can move up, down, right, left, stationary

(alphabet written from input set based on state of mind)

Convert this 2D into 1D (number of cols fixed, no of rows infinite).

now direction in 1D → right, left, stationary



(sequential access):

head/pointer

(move right, stationary)

if left → crash

- In FA, PDA decision was partial as input was online
- In Turing offline input so complete decision.
- Halt means taking decision after complete processing of string

halt: h_a (accept), h_r (reject)

rejection: explicit (h_r), implicit (crash)

Date:

- Input is always enclosed b/w Δ in the tape

eg x

#	Δ	x	Δ

tape

eg x = aba

#	Δ	a	b	a	Δ	......

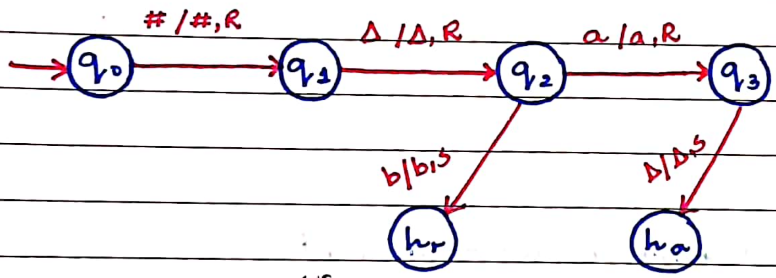
infinite Δ's

$L_1 = \{a\}$

#	Δ	a	Δ	...

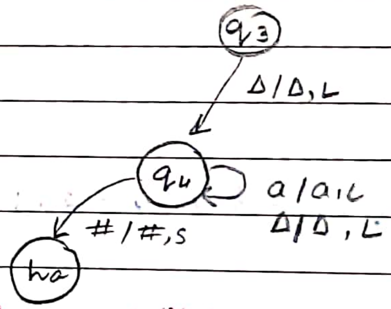
current element ← #/#,R → Direction of head

change ↑



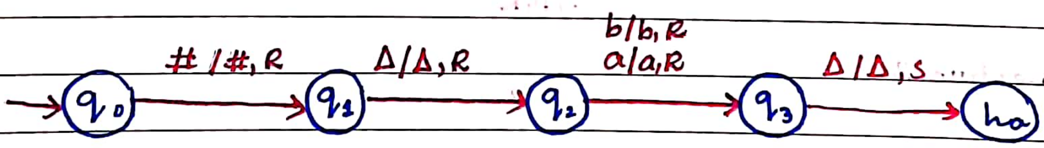
(if showing ~~explicit~~ rejection)

(if requirement to bring head on #)

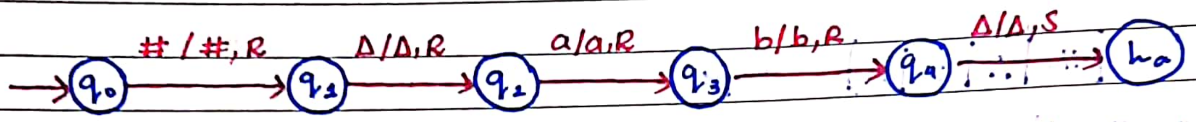


$\Sigma = \{a,b\}$

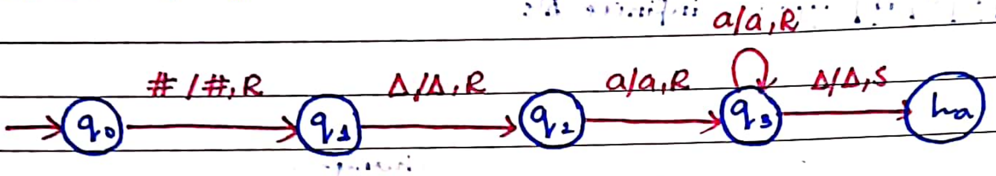
$L_2 = \{a,b\}$



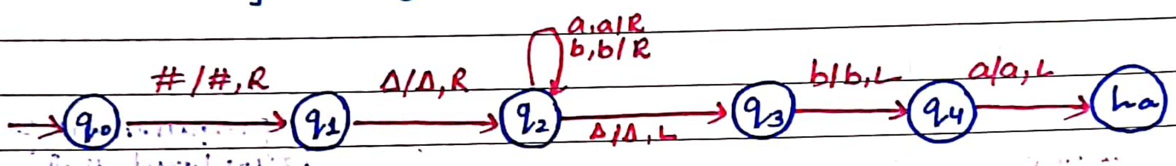
→ $L_3 = \{ab\}$



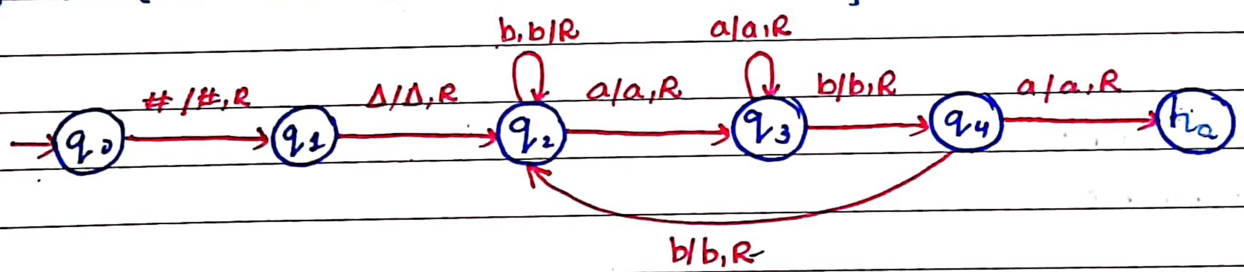
→ $L_4 = \{a^+\}$



→ L5 : String ending with ab



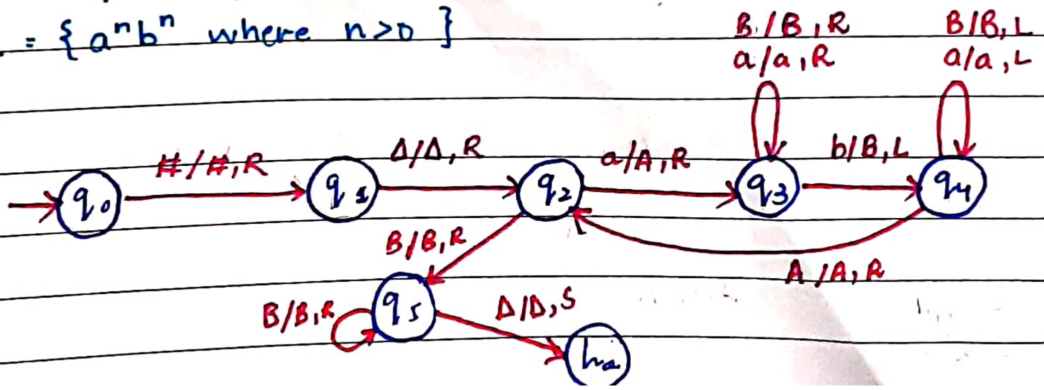
→ $L_6 = \{ x \mid x \in \{a,b\}^* \text{ and } x \text{ has a substring } aba \}$



→ Week 13 : Lecture 2

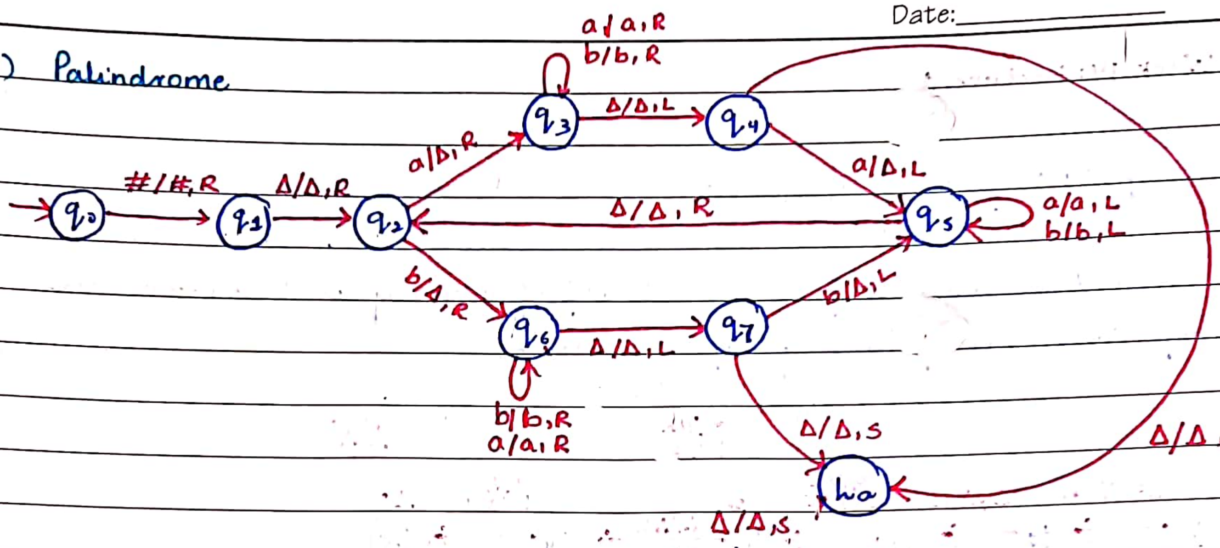
Develop TM for

i) $L = \{a^n b^n \text{ where } n > 0\}$



Date:

ii) Palindrome



Homework

$L_1 = \{a^n b^n c^n$, where $n \geq 0\}$

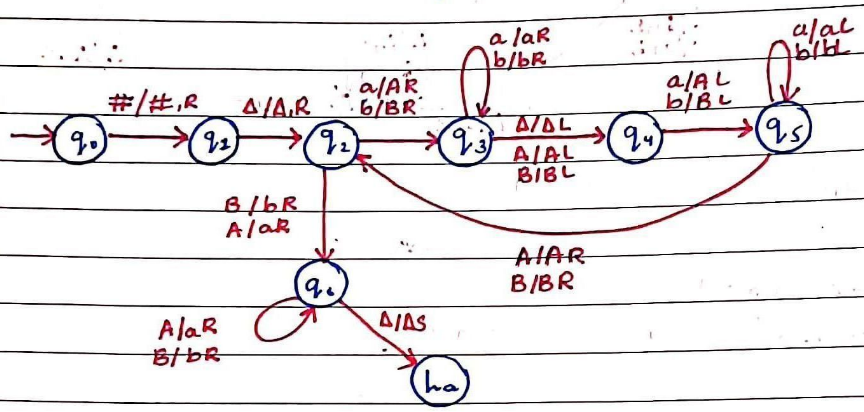
$L_2 = \{a^n b^{2n}$, where $n \geq 0$

Date:

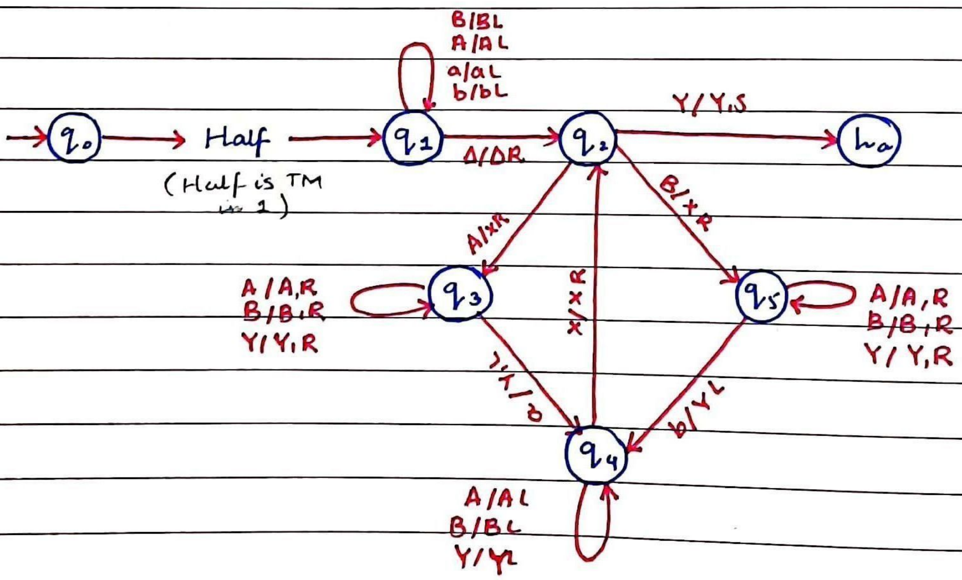
→ Week 14: Lecture 1

Develop TM for $\Sigma = \{a, b\}$

1. Should be able to find the half of string and convert one half into corresponding capital letter.



2. $L_2 = \{xx$ where $x \in \Sigma^*\}$



Scanned with CamScanner

Date:

→ Week 14: Lecture 2

Tuning Machine with more than 1 input

$f(x,y) = x \text{ OR } y$

$|x| = |y|$

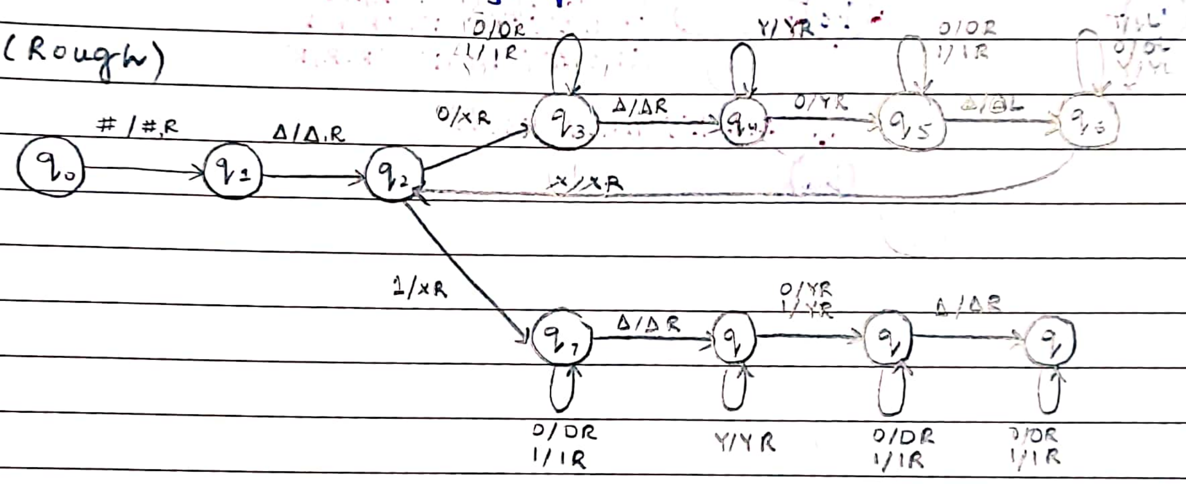
Compute $f(x,y)$ with tuning machine

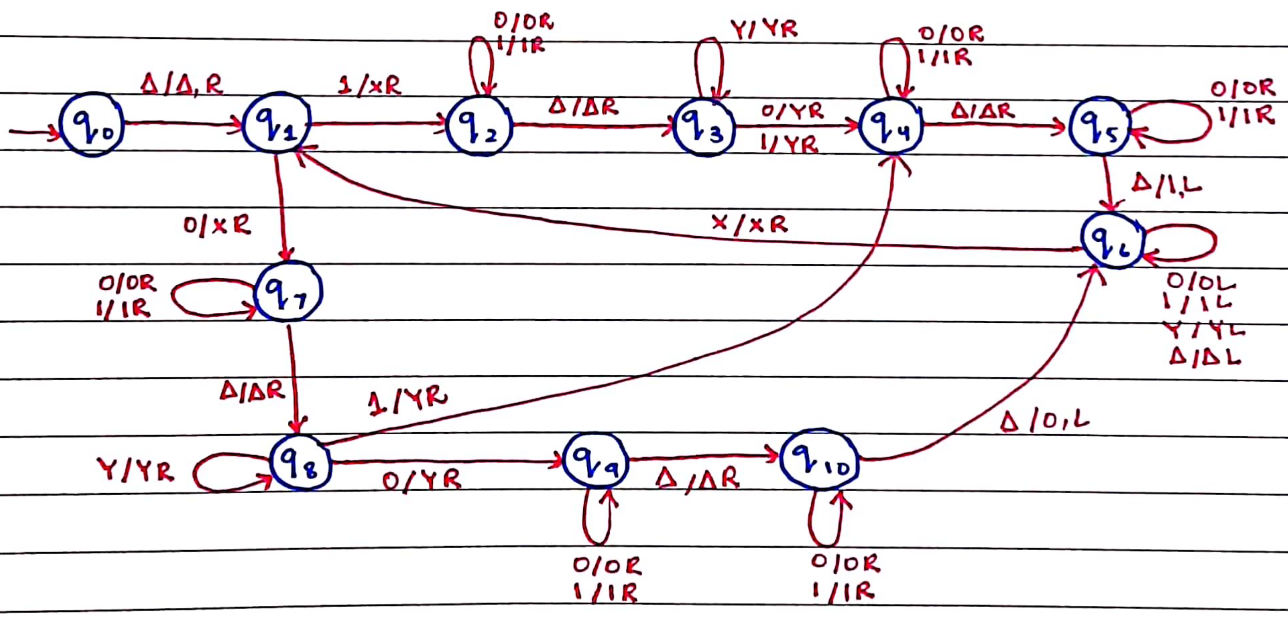
#	Δ	x	Δ	y	Δ	x OR y	Δ

$x, y \in \{0,1\}^*$

- We assume length of x and y equal

(Rough)



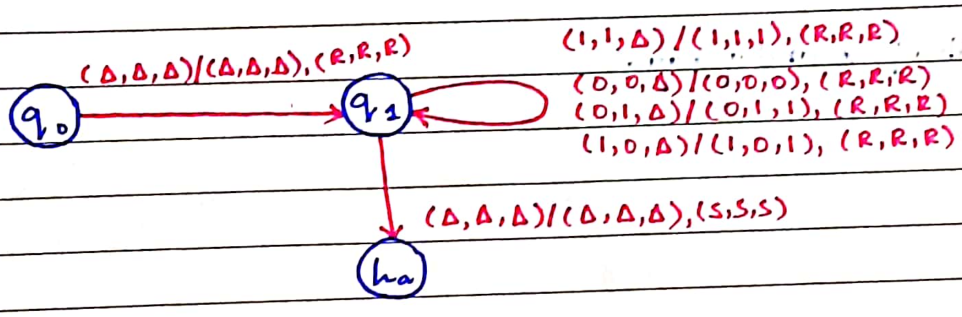


Date:

Multitape Turing Machine

3 tape turing machine
input + output

Δ	1	0	1	Δ
Δ	0	0	1	Δ
Δ	1	0	1	Δ



⇒ Week 15: Lecture 1

Languages
- Recursive
- Recursively enumerable

Computes a function

$$f(x) = \begin{cases} 1, & \text{accept} \\ 0, & \text{reject} \end{cases}$$

halt
- accept h_a
- reject h_r

accept → h_a

halt
↓
accept

explicit acceptance

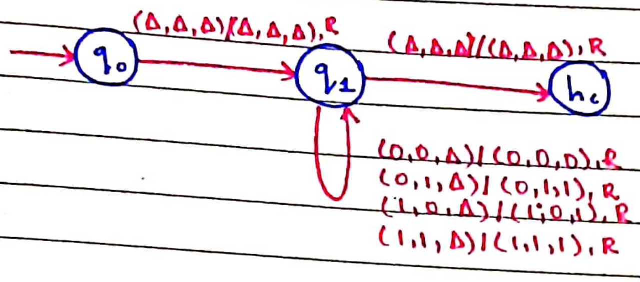
rejection may be explicit or implicit

every Rec is RE but every RE isn't Rec

Multitrack

multiple tape with 1 head

X OR Y

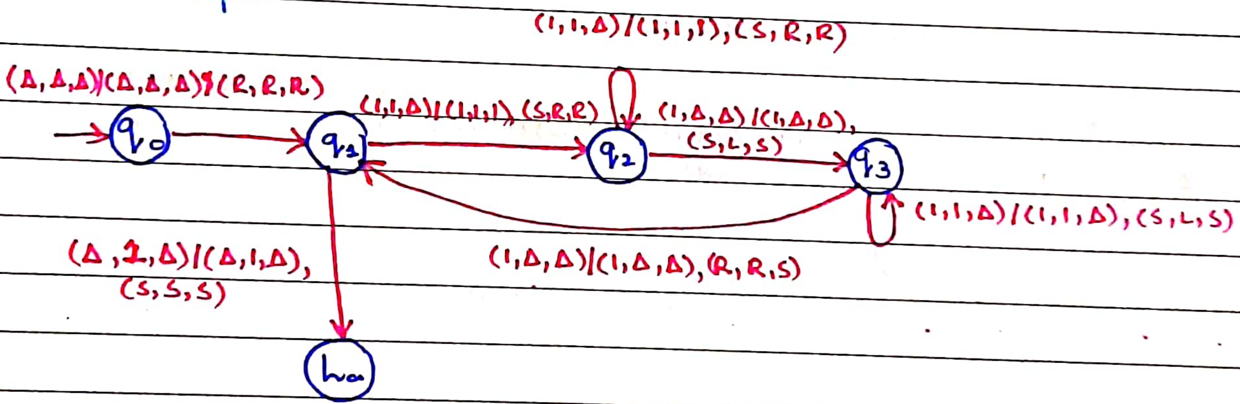


function compute
Recursive

Develop 3 Tape TM with 2 inputs

tape 1	Δ	1	1	Δ	Δ	Δ	Δ
tape 2	Δ	1	1	1	Δ	Δ	Δ
tape 3	Δ	1	1	1	1	1	1

Multitape



Date:

→ Week 15: Lecture 2

Q. Develop 3-tape multitape TM that takes 2 binary numbers x and y. |x| = |y|. Perform addition and place the result on 3rd tape.

x → tape 1

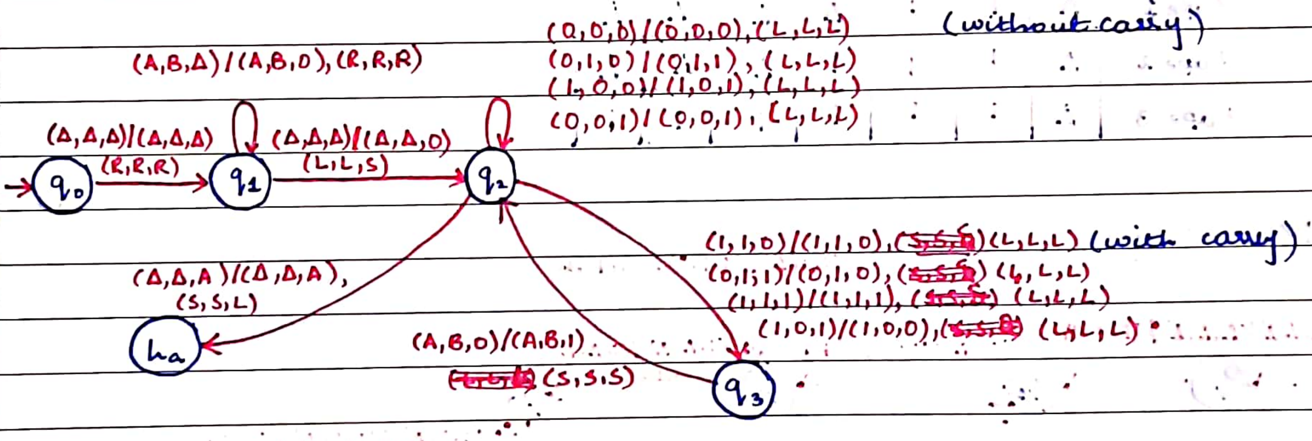
y → tape 2

x+y → tape 3

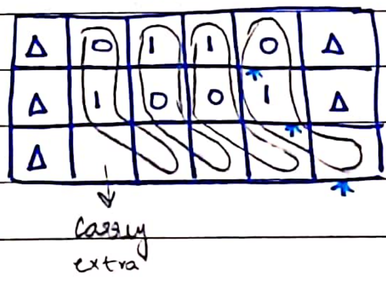
A = {0,1}

B = {0,1}

Multitape



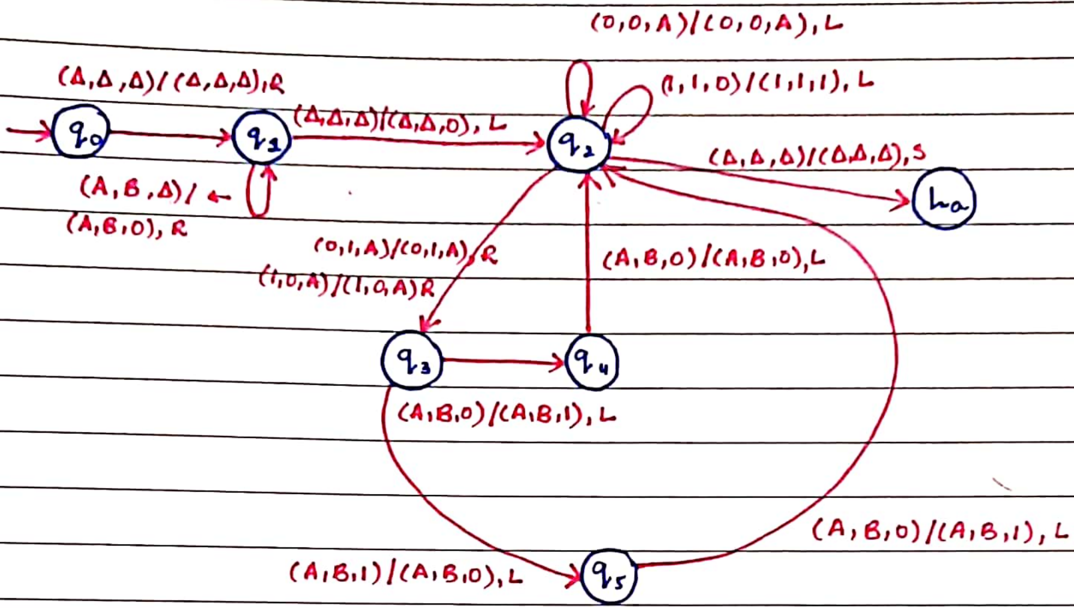
x	y	z	s	c	
0	0	0	0	0	no carry
0	0	1	1	0	no carry
0	1	0	1	0	no carry
0	1	1	0	1	
1	0	0	1	0	no carry
1	0	1	0	1	
1	1	0	0	1	
1	1	1	1	1	



Scanned with CamScanner

Date:

Multitrack



Scanned with CamScanner