Theory Of Automata

Date: _____

What does automata mean?
It is the plural of automation, and it means "something that works autom

There are two types of languages.

•) Formal Language (Syntactic language)
↳ concerned with rules & meaning
Eg: In Eng lang we write work, not wkro

•) Informal Language (Semantic Language
↳ concerned with rule & syntax not with meaning
Eg: Lang1: Start with w & end with o so wkro is right not work

A finite non-empty set of symbols (letters) is called an alphabet. It is denoted by Σ (Greek letter sigma)
Example: $\Sigma = \{a, b\}$, $\Sigma = \{0, 1\}$, $\Sigma = \{i, j, k\}$

A certain version of language ALGOL has 113 letters.
Σ (alphabet) includes letters, digits & variety of operators including sequential operator such as GOTO and IF

Concatenation of finite symbols from the alphabet is called a string
Example: if $\Sigma = \{a, b\}$ then : a, abab, abababababab.

Empty String or Null String: Sometimes a string with no symbol at all is used denoted by λ or Λ is called an empty string or null string.

Words are strings belonging to some language.
Example: If $\Sigma = \{x\}$ then a language L can be defined as
$L = \{x^n : n = 1, 2, 3 \cdots\}$ or $L = \{x, xx, xxx, \cdots\}$

All words are strings, but not all strings are words.

Date: _____

Valid/In-Valid alphabets:

While defining an alphabet, an alphbet may contain letters consisting of group of symbols for example $\Sigma_1 = \{B, aB, bab, d\}$
Now consider an alphabet
$\Sigma_2 = \{B, Ba, bab, d\}$ and a string BababB.

As when this string is scanned by the compiler (lexical Analyzer) first symbol B is identified as a letter belonging to Σ, while for the second letter the lexical analyzer would not be able to identify, so while defining an alphabet it should be kept in mind that ambiguity should not be created.

While defining an alphabet of letters consisting of more than one symbols no letter should be started with the letter of same alphabet i.e one letter should not be the Prefix of another. However a letter may be ended in the letter of same alphabet i.e one letter may be the suffix of another.

$\Sigma_1 = \{B, aB, bab, d\}$ $\Sigma_2 = \{B, Ba, bab, d\}$

Σ_1 is a valid alphabet while Σ_2 is an in-valid alphabet.

The length of string s, denoted by $|s|$, is the number of letters in the string.

Example: $\Sigma = \{a, b\}$, $s = ababa$
$|s| = 5$

$\Sigma = \{B, aB, bab, d\}$, $s = BaBba$
$|s| = 4$

Regular Expression: Lang is represented in terms of strings.

Eg: L = [^, a, aa, aaa, ...]
So, R = a*

Eg: L = [a, aa, aaa, ...]
R = a+

Eg: L = [a, ab, abb, ...]
R = ab* → Concatenation of String (AND)

Eg: string contain a or b
L = [a, b]
R = a+b → Union string + OR

Eg: String contain any a or any b
R = (a+b)*

Eg: start with a and end a
R = a(a+b)*a

(1) Define a RE for lang that contain substring ba
(a+b)* ab (a+b)*

(2) All string which do not contain substring ba.
b*a* ✗ a*b* ✓

(3) All strings which don't contain substring 00
0+1*+1*01*

(4) Don't contain 101
0*(1*000*)*1*0*

Power of String
ba²b = baab
ba*b = bb, bab, ...
ba+b = bab, baab ...

Date:

Reverse of a string

The reverse of string s denoted by Rev(s) or s^r is obtained by writing the letters of s in reverse order.

if s = abc then Rev(s) = cba. s = BaBbabBd s^r = dBbabaBB

The languages can be defined in different ways, such as Descriptive definition, Recursive definition, using Regular Expression (RE) and using Finite Automation (FA) etc.

Descriptive definition of language: The language is defined, describing the conditions imposed on its words.

Example: The language L of strings of odd length, defined over Σ = {a}
canbe written as L = {a, aaa, aaaaa...}

The language L of strings that doesnot start with a defined over Σ = {a, b, c}
canbe written as L = {b, c, ba, bb, ca, cb ....}

L of string length 2, Σ = {0, 1, 2} L = {00, 01, 11, 02, 12, 21, 22, 20}

L of string ending in 0, Σ = {0, 1} L = {0, 00, 10, 000, 010, 100, 110 ....}

The language EQUAL of strings with number of a's equal to b's
canbe written as: {Λ, ab, aabb, abab, abba ....}

The language EVEN-EVEN, Σ = {a, b}, RE = [aa+bb+(ab+ba)(aa+bb)*(ab+ba)]*
written as {Λ, aa, bb, aabb, abab, baab, baba ....}

The language INTEGER, Σ = {-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
INTEGER = {..., -2, -1, 0, 1, 2, ...}
EVEN = {..., -4, -2, 0, 2, 4}

Language $\{a^n b^n\}$, of string defined over Σ = {a, b}
$\{a^n b^n : n = 1, 2, 3, \ldots\}$ can be written {ab, aabb, aaabbb, aaaabbbb}

Language $\{a^n b^n a^n\}$
{aba, aabbaa, aaabbbaaa, aaaabbbbaaaa}

(1) Define RE for even No. of a.
b* + (b*ab*ab*)*

Date: _____

The language factorial, $\Sigma = \{1,2,3,4,5,6,7,8,9\}$
i.e $\{1, 2, 6, 24, 120, \cdots\}$
Language FACTORIAL, $\Sigma = \{a\}$ as $\{a^{n!} : n = 1,2,3, \cdots\}$
$\{a, aa, aaaaaa, \cdots\}$ FACTORIAL canbe defined over any single letter alphabet
Double FACTORIAL, $\Sigma = \{a, b\}$, as $\{a^{n!} b^{n!} : n = 1,2,3 \cdots\}$
$\{ab, aabb, aaaaaabbbbbb\}$
SQUARE, $\Sigma = \{a\}$ as $\{a^{n^2} : n = 1,2,3 \cdots\}$
$\{a, aaaa, aaaaaaaaa, \cdots\}$
DOUBLESQUARE, $\Sigma = \{a,b\}$, as $\{a^{n^2} b^{n^2} : n = 1,2,3 \cdots\}$
$\{ab, aaaabbbb, aaaaaaaaabbbbbbbbb\}$.
PRIME, $\Sigma = \{a\}$ as $\{a^p : p \text{ is prime}\}$
$\{aa, aaa, aaaaa, aaaaaaa, \cdots\}$

PALINDROME: The language consisting of Λ and the strings s defined over Σ such that $Rev(s) = s$.
Example: For $\Sigma = \{a, b\}$
PALINDROME $= \{\Lambda, a, b, aa, bb, aaa, aba, bab, \cdots\}$

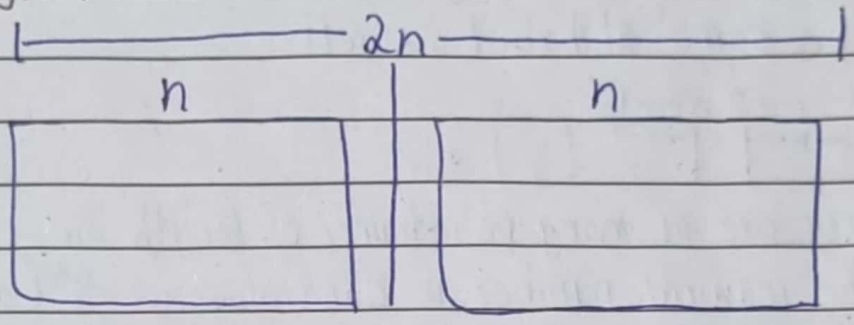
There are as many palindromes of length 2n as there are of 2n-1

Number of strings of length 'm' defined over alphabet of 'n' letters is n^m
Example: language of strings of length 2, $\Sigma = \{a,b\}$ is $L = \{aa, ab, ba, bb\}$ i.e 2^2.
Length = 3, $\Sigma = \{a,b\}$ $L = \{aaa, aab, aba, baa, baa, abb, bab, bba, bbb\}$
i.e number of strings $= 2^3$

2n → results in an even number.

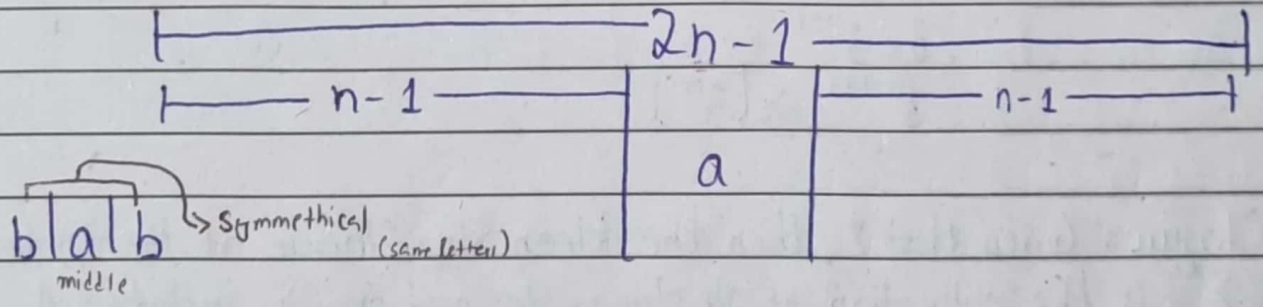
Date: _____

To calculate the number of palindromes of length (2n), consider the following diagram.



which shows that there are as many palindromes of length 2n as there are the stings of length n i.e the required number of palindromes are 2^n

To calculate the number of palindrome of length (2n-1) (odd) with 'a' as the middle letter, consider the following diagram.



2n	2n - 1	extracting the middle part
n=1 n=2		2n - 1(-1)
2(1) = 2	2(1) - 1 = 1	2n - 2
2(2) = 4	2(2) - 1 = 3	2 (n-1)

↳ represent the 2 symmetrical part.

which shows that the are as many palindromes of length 2n-1 as there are the strings of length n-1 i.e the required number of palindrome are 2^{n-1}
Similarly the number of palindromes of length 2n-1, with 'b' as middle letter will 2^{n-1} also
Hence the total number of palindrome of length 2n-1 will be $2^{n-1} + 2^{n-1} = 2(2^{n-1})$
$\therefore = 2^n$

Date: _____

Q) Prove that there are as many palindromes of length 2n, defined over $\Sigma = \{a, b, c\}$ as there are of length 2n-1. Determine the number of palindromes of length 2n defined over the same alphabet as well.

even — 2n

For length 2n: n | n | 3^n

which shows that there are as many palindromes of length 2n as there are strings of length n i.e the required number of palindrome are 3^n (as there are 3 letters in the given alphabet so the number of strings of length n will be 3^n).

$\boxed{3^n}$

For length 2n-1 with a as middle letter: n-1 | a | n-1-1

n-1 → length of string, 3 → no of alphs ∴ 3^{n-1}

a b c

$3^{n-1} + 3^{n-1} + 3^{n-1}$

$3(3^{n-1})$

$3 \times 3^{n-1}$

$3^{n-1+1} \Rightarrow \boxed{3^n}$

Kleene Star Closure: Given ~~that~~ Σ, then the kleen Star Closure of the alphabet Σ denoted by Σ^*, is the collection of all strings defined over Σ, including Λ. It is noted that kleene star closure can defined over any set of strings.

if $\Sigma = \{x\}$ then $\Sigma^* = \{\Lambda, x, xx, xxx, xxxx, \dots\}$

if $\Sigma = \{0, 1\}$ then $\Sigma^* = \{\Lambda, 0, 1, 00, 01, 10, \dots\}$

if $\Sigma = \{aaB, c\}$ then $\Sigma^* = \{\Lambda, aaB, c, aaBaaB, aaBc, \dots\}$

Languages generated by kleene Star Closure of set of strings are infinite language (By infinite language, it is supposed that the language contains infinite many words, each of finite length).

Date: _____

Plus Operation $(^+)$ is same as kleene Star Closure except that it does not generate Λ (null string), automatically.

IF $\Sigma = \{0, 1\}$ Then $\Sigma^+ = \{0, 1, 00, 01, 10, 11, \cdots\}$

IF $\Sigma = \{aab, c\}$ Then $\Sigma^+ = \{aab, c, aabc, aabaab, caab \cdots\}$

(Q) Is there any case when S^+ contains Λ? If Yes then justify your answer.

Let $S = \{\Lambda, a, b\}$

$S^+ = \{\Lambda, a, b, aa, ab, ba, \cdots\}$

S^+ contains the Λ if & only if S itself contains the empty string.

(Q) Prove that for any set of strings S. $S = \{a, b\}$

(i) $(S^+)^* = (S^*)^*$

$S^+ = \{a, b, ab, ba, \cdots\}$

$S^* = \{\Lambda, a, b, ab, ba, \cdots\}$

$(S^+)^* = \{\Lambda, a, b, ab, ba, \cdots\}$

$(S^*)^* = \{\Lambda, a, b, ab, ba, \cdots\}$

$\therefore (S^+)^* = (S^*)^*$

(ii) $(S^+)^+ = S^+$

$S^+ = \{a, b, ab, ba, \cdots\}$

$(S^+)^+ = \{a, b, ab, ba, \cdots\}$

$\therefore (S^+)^+ = S^+$

(iii) Is $(S^*)^+ = (S^+)^*$

Yes!

Finite Automata Date: _____

Finite automata are finite collections of states with transition rules that take you from one state to another.

Original application was sequential switching circuits, where the "state" was the setting of internal bits

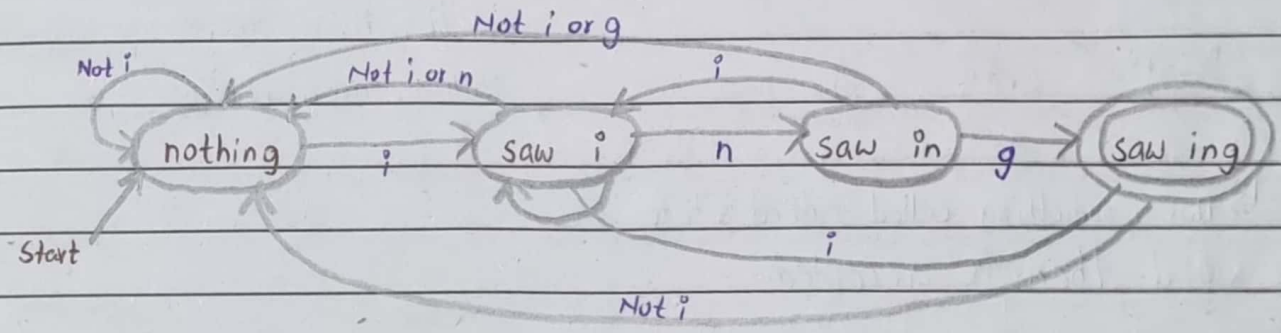
Today, several kinds of software can be modeled by Finite Automata.

Representing Finite Automata.

Simplist representation is often a graph

- Nodes = states
- Arcs indicates state transitions
- Labels on arcs tell what causes the transition.

Example: Recognizing Strings Ending in 'ing'



Automata to Code

(i) Initialize state q to start state

(ii) Loop through the string one character at a time.

(iii) Make a switch statement with a case for each state for q, where each case sets q according to the transitions for that state.

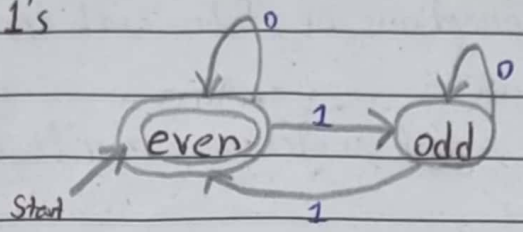
(iv) Accept if you end in a final state.

Example in Java:

```
Scanner scan = new Scanner(System.in)
String s = scan.next()
int q = 0 ← start state.
for(char c : s.toCharArray()){ ← Loop through string s
  switch(q){                                   Transition ↔
    case 0: q(c == 'i') ? 1 : 0; break;
    case 1: q(c == 'n') ? 2 : ((c == 'i') ? 1 : 0); break;
    case 2: q(c == 'g') ? 3 : ((c == 'i') ? 1 : 0); break;
    case 3: q = (c == 'i') ? 1 : 0; }}
if (q == 3) → Final State.
  S.out("accept");
else sout("reject");
```

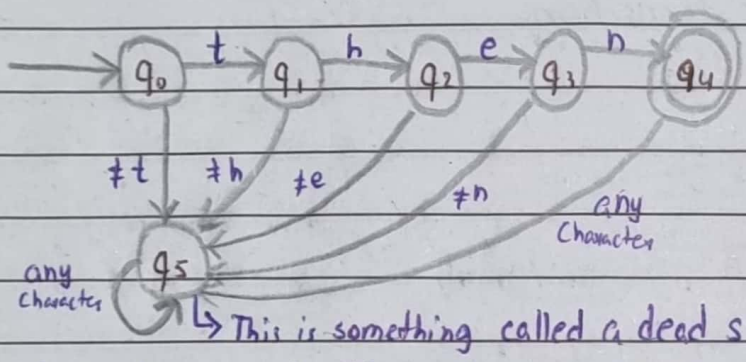
Date: _____

Example: An even Number of 1's



) How would it look to accept a number of 1's that is multiple of 3?

Password/keyword Example:

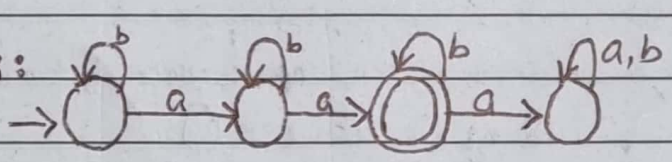


↳ This is something called a dead state.

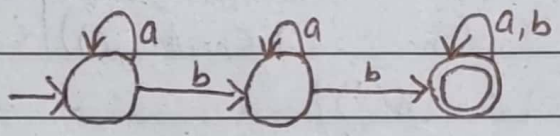
~~Oth~~ Only the word then is accepted.

Btw, there is a potential security risk on the password application if this finite automaton reports failure too quickly.

Exactly Two a's:

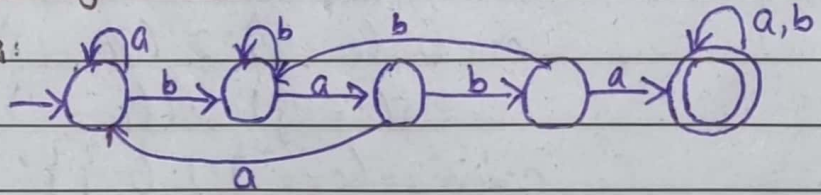


Atleast Two b's:

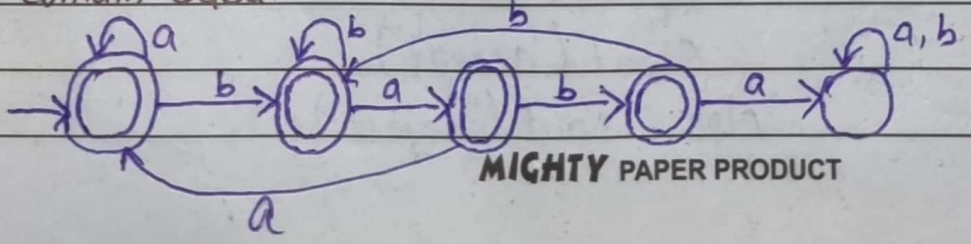


Containing Substrings or Not

Contains baba:



Does not contain baba:



Sometime things are easy with finite automata
- Substrings (....abcaba....)
- Subsequence (...a....b....c....b....a...)
- Modular Counting (odd number of 1's)

Something are impossible with finite automata
-) More 0's than 1's
-) An equal number of a's and b's

But when they can be used, they are fast.

Date: _____

Alphabets:
An alphabets is any finite set of symbols.
Exmp: ASCII, Unicode, {0,1} (binary alphabet), {a,b,c}

Strings: The set of strings over an alphabet Σ is the set of lists, each element of which is a member of Σ
-) Sting shown with no commas, e.g abc.
•) Σ^* denotes this set of strings.
•) ε stands for the empty string (string of length 0).
Eg: $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 000, 001, \cdots\}$

Language: is a subset of Σ^* for some alphabet Σ.
Eg: The set of strings 0's and 1's with no two consecutive 1's
$L = \{\varepsilon, 0, 1, 00, 01, 10, 001, 010, 101, 0010, 0101, 1001, 1010, \cdots\}$

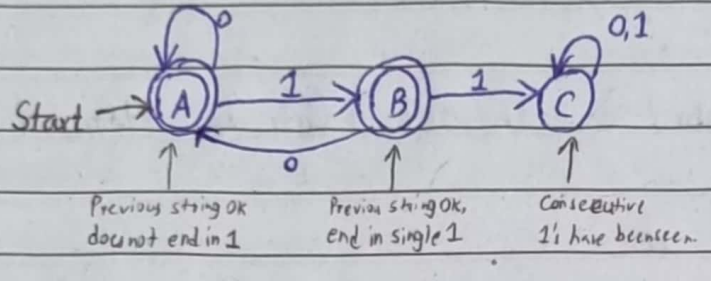
↱ Specific output (one output) ↱ $q_3 \to q_4$
↱ if you apply any alphabet, so it will take you to specific state
Deterministic Finite Automata: A Formalism for defining languages, consisting of:
1) A finite set of states (Q, typically) 2) An input alphabet (Σ, typically) → (a, b)
3) A transition function (δ, typically) 4) A start state (q_0, in Q, typically).
5) A set of final states ($F \subseteq Q$, typically) "Final" and "accepting" are synonyms.
↳ $F \subseteq Q$

The Transition Function:
•) Take two arguments: a state and an input symbol.
•) $\delta(q, a)$ = the state that the DFA goes to when it is in state q and input a is received

Date: _____

Graph of an DFA:

Accepts all strings without two consecutive 1's



Alternative Representation:

Transition Table

	0	1
→*A	A	B
*B	A	C
C	C	C

← Columns = input symbols.

↑ Row = states

- Final state starred.
- Arrow for start state.

Extended Transition Function:

We describe the effect of a string of inputs on a DFA by extending δ to a state and a string.

- Induction on length of string.

Basis: $\delta(q, \epsilon) = q$ Induction: $\delta(q, wa) = \delta(\delta(q, w), a)$

w is a string; a is an input symbol.

Eg: Extended Delta

	0	1
A	A	B
B	A	C
C	C	C

$\delta(B, 011) = \delta(\delta(B, 01), 1) = \delta(\delta(\delta(B, 0), 1), 1) \Rightarrow$

$\delta(\delta(A, 1), 1) \Rightarrow \delta(B, 1) \Rightarrow C$

Delta-hat

Some people denote the extended δ with "hat" to distinguish it from δ itself

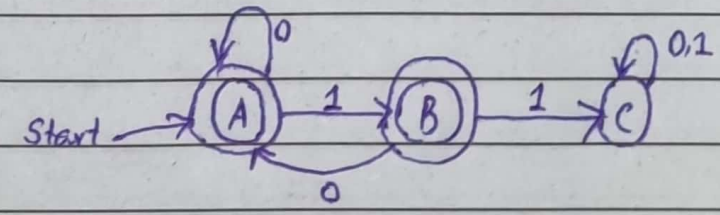
Not needed cuz both agree when the string is a single symbol.

$\hat{\delta}(q, a) = \delta(\hat{\delta}(q, \epsilon), a) = \delta(q, a)$

Extended Delta.

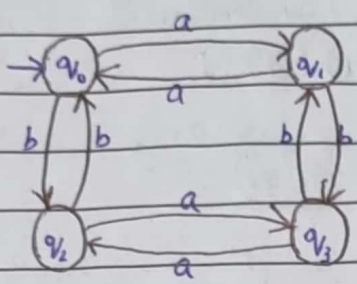
Sting in a Language of a DFA

$\{w \mid w \text{ is in } \{0,1\}^* \text{ and } w \text{ does not have two consecutive 1's}\}$



DFA Questions:

Date: _____



If we make final state as following:

q_0: Even a & Even b

q_1: Odd a & Even b

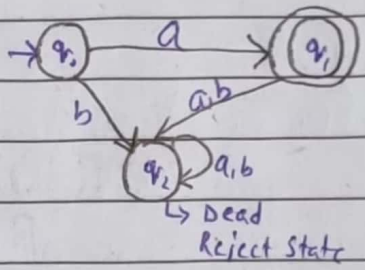
q_2: Even a & odd b

q_3: Odd a & odd b

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

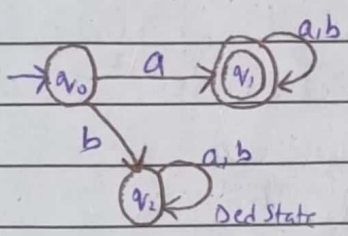
$R = [aa + bb + (ab + ba)(aa + bb)^*(ab$

Q) Draw DFA that accepts exactly a. $R = a$



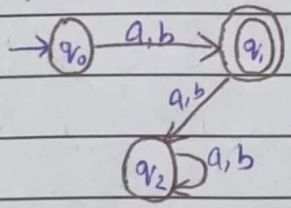
Q) Accept all words starting with a.

$R = a(a+b)^*$



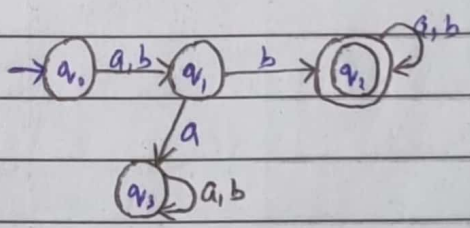
Q) Accept exactly a or b.

$R = a + b$



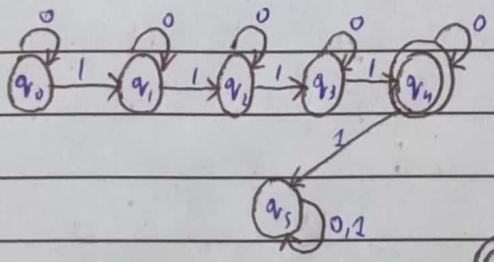
Q) that have b as second letter over $\Sigma = \{a, b\}$

$R = (a+b)\,b\,(a+b)^*$



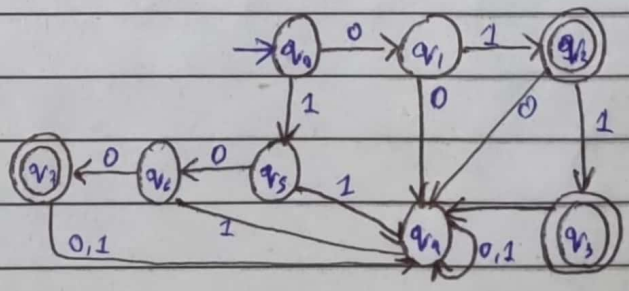
Q) Have exactly 4 ones in every string $\Sigma = \{0, 1\}$

$R = 0^*10^*10^*10^*10^*$



Q) That only accept $L = \{01, 011, 100\}$ over $\Sigma = \{0$

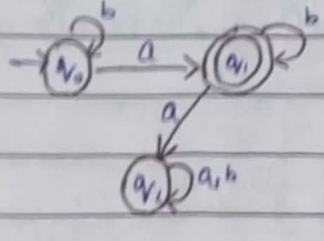
$R = 01 + 011 + 100$



Date: _____

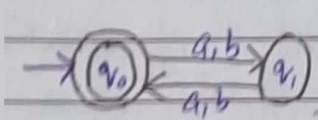
① Draw for $L = \{w : n_a = 1 \; w \in \{a,b\}\}$

$R = b^* a b^*$



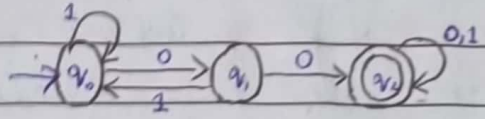
② Accept all string with even length $\Sigma = \{a,b\}$

$R = (aa + ab + ba + bb)^*$



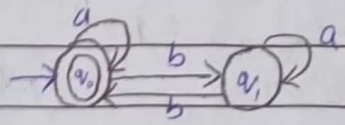
③ $L = \{w \in \{0,1\}^* \mid w \text{ contain } 00 \text{ as substring}\}$

$R = (0+1)^* 00 (0+1)^*$



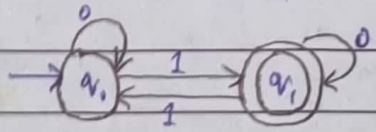
④ Even no. of b's $\Sigma = \{a,b\}$

$R = a^* + (a^* b a^* b a^*)^*$



⑤ Odd no. of 1's $\Sigma = \{0,1\}$

$R = 0^* 1 0^* (1 0^* 1 0^*)^*$



Date: _____

NFA $(Q, \Sigma, q_0, F, \delta)$ → Regular Lang

finite state

0,1 a,b

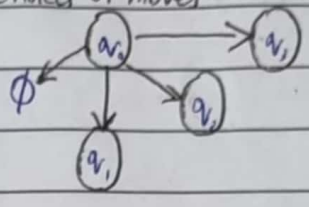
$F \subseteq Q$

$\delta : Q \times \Sigma \Rightarrow Q$

$(q_0, q_1, q_2) \times (0,1)$ 2^0

$2^3 = 8$

Choices of moves



*) one letter can move to more than one state, next state is not determ

*) Can move to other state without any input i.e ε/null.

q_0 0	ϕ	
q_0 1	q_0	
q_1 0	q_1	
q_1 1	q_2	
q_2 0	$q_0 q_1$	
q_2 1	$q_0 q_2$	
	$q_1 q_2$	
	$q_0 q_1 q_2$	

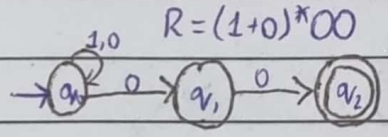
Major difference b/t DFA & NFA is δ (transition)

DFA	NFA
1) Dead configration is NOT allowed	Dead configration is allowed.
2) Multiple choices aren't available	Multiple choices are available corresponding to in
3) ε-Move is not allowed	ε-move is allowed.
4) Digital computers are deterministic	ND features is not associated with real com
5) Designing & understanding difficult	Designing & understanding is easy.

NFA → DFA, 2^n possible states.

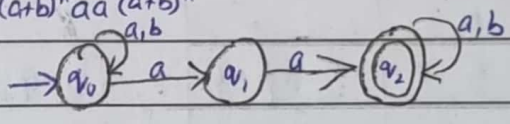
① NFA end with 00, $\Sigma = \{0,1\}$

$R = (1+0)^*00$



② Accept all words contain substring aa $\Sigma = \{a,b\}$

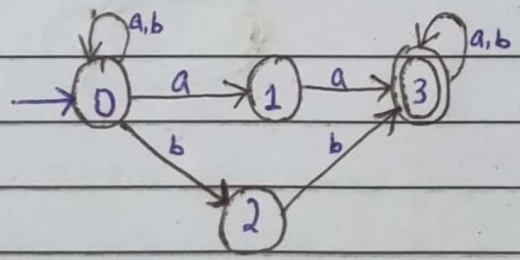
$R = (a+b)^* aa (a+b)^*$



Non-Determinism: finite automaton has the ability to be is several states at on
Transitions from a state on an input symbol can be to any set of states.
State in one state - Accept if any sequence of choices leads to a final state.
Intuitively: the NFA always "guesses right".

Every NFA has a DFA
↳ DFA exist for all NFA
↳ Proof is subset constructor.

Example: 2 consecutive a's & b's

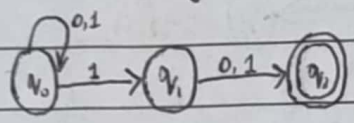


No. of states of DFA can be exponential in the no. of states of NFA

Analogy: a class of objects whose values are sets of objects of another class.

Date: _____

Q) NFA of all binary strings which 2nd last bit is 1.



NFA - Transition table

	States	0	1
→	q_0	q_0	q_0, q_1
	q_1	q_2	q_2
	q_2	-	-

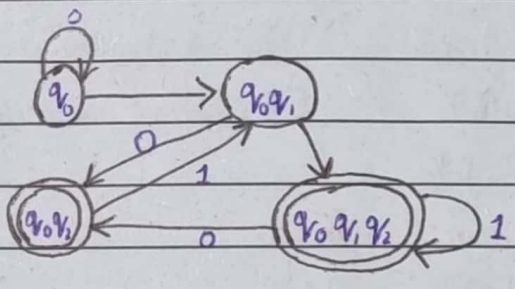
3 states ⇒ $2^3 = 8$
↳ Max Possible State of DFA

DFA Transition table using NFA table

	States	0	1
→	q_0	q_0	$q_0 q_1$
	$q_0 q_1$	$q_0 q_2$	$q_0 q_1 q_2$
*	$q_0 q_2$	q_0	$q_0 q_1$
*	$q_0 q_1 q_2$	$q_0 q_2$	$q_0 q_1 q_2$

↳ both are final states as both includes q_2.

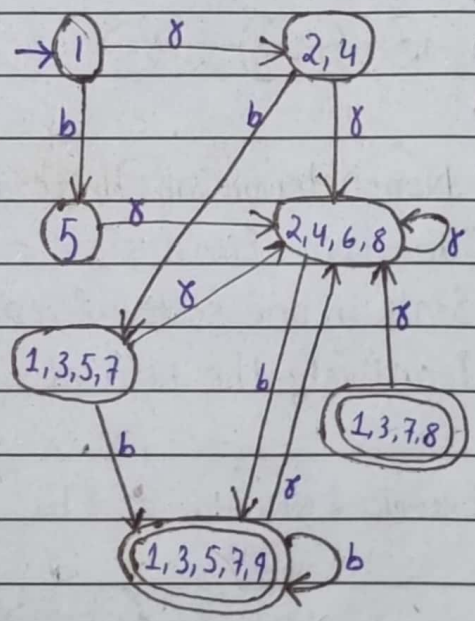
DFA



Q) We will construct DFA equivalent to our "chessboard" NFA

	a	b
→ 1	2,4	5
2	4,6	1,3,5
3	2,6	5
4	2,8	1,5,7
5	2,4,6,8	1,3,7,9
6	2,8	3,5,9
7	4,8	5
8	4,6	5,7,9
* 9	6,8	5

	a	b
→ {1}	{2,4}	{5}
{2,4}	{2,4,6,8}	{1,3,5,7}
{5}	{2,4,6,8}	{1,3,7,9}
{2,4,6,8}	{2,4,6,8}	{1,3,5,7,9}
{1,3,5,7}	{2,4,6,8}	{1,3,5,7,9}
*{1,3,7,9}	{2,4,6,8}	{5}
*{1,3,5,7,9}	{2,4,6,8}	{1,3,5,7,9}

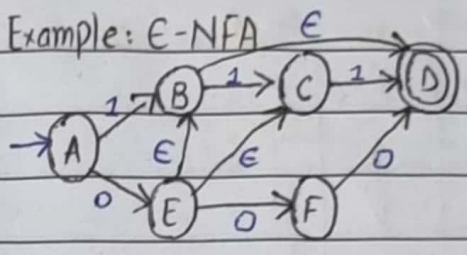


Date: _____

NFA's with ε-Transitions:

We allow state-to-state transition on ε input.

Example: ε-NFA



Closure of state: $CL(q_i)$

Eg: $CL(A) = \{A\}$;

$CL(E) = \{B, C, D, E\}$

Closure of set of states = Union of closure each state

Extended Delta:

Basis: $\hat{\delta}(q, \epsilon) = CL(q)$

Eg: $\hat{\delta}(A, \epsilon) = CL(A) = \{A\}$

$\hat{\delta}(A, 0) = CL(\{E\}) = \{B, C, D, E\}$

$\hat{\delta}(A, 01) = CL(\{C, D\}) = \{C, D\}$

ε-NFA

	0	1	ε
→ A	{F}	{B}	∅
B	∅	{C}	{D}
C	∅	{D}	∅
* D	∅	∅	∅
E	{F}	∅	{B, C}
F	{D}	∅	∅

Since closure of B and E include final state D

ε-NFA to NFA

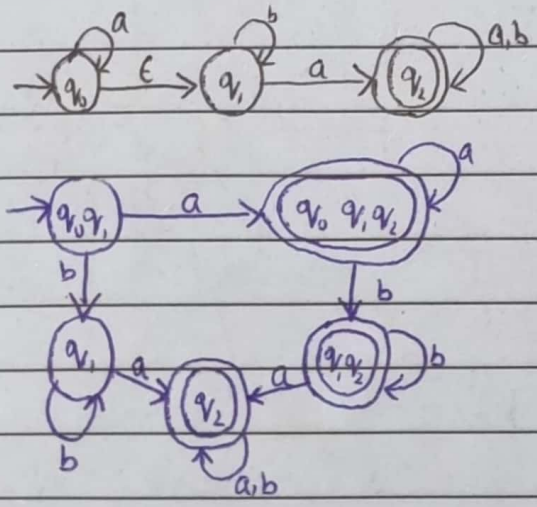
	0	1
→ A	{E}	{B}
* B	∅	{C}
C	∅	{D}
* D	∅	∅
* E	{F}	{C, D}
F	{D}	∅

Since closure of E includes B and C; which have transitions on 1 to C a

Every NFA is an ε-NFA → its just no transitions on ε

Converse requires us to take ε-NFA & construct NFA that accept same lang.

ε-NFA to DFA



ε closure of $q_0 = q_0q_1 \xrightarrow{a} a(q_0) \cup a(q_1)$

$q_0q_1 \cup a(q_2) = q_0q_1q_2$

$q_1 \xrightarrow{a} q_2$, $q_2 \xrightarrow{b} q_2$

$q_0q_1 \xrightarrow{b} b(q_1) \cup b(q_1) = q_1$

$q_0q_1q_2 \xrightarrow{a} q_0q_1q_2 \cup q_2 = q_0q_1q_2$

$q_0q_1q_2 \xrightarrow{b} q_1 \cup q_2 = q_1q_2$

$q_1q_2 \xrightarrow{a} q_2 \cup q_2 = q_2$

$q_1q_2 \xrightarrow{b} q_1 \cup q_2 = q_1q_2$

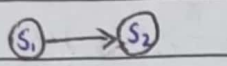
DFA, NFA, ε-NFA all accept exactly same set of lang: the regular lang.

But only DFA can be implemented in linear time.

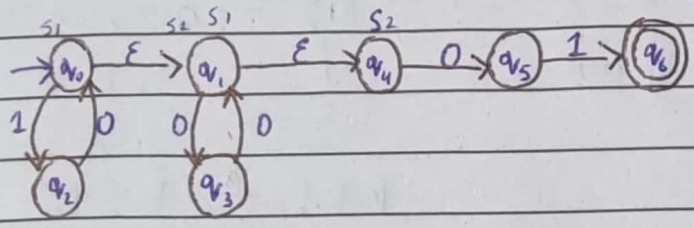
ε NFA States = NFA States
Same.

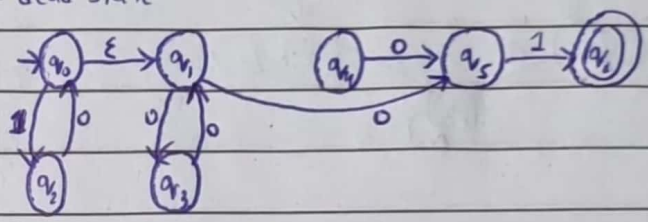
Date:

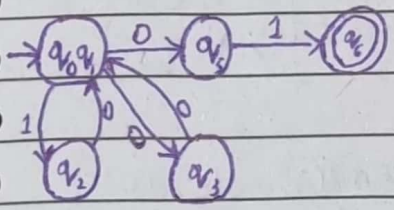
ε-NFA → Eliminates ε-Moves:

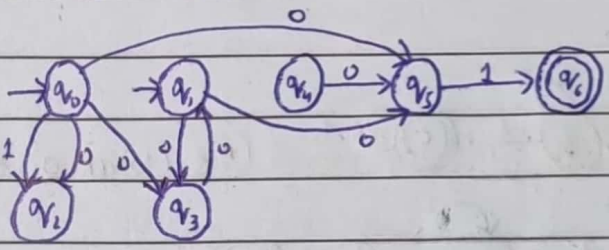


1) Find all edges starting from S_2.
2) Duplicate all edges to S_1 without changing edge labe
3) If S_2 is initial state, make S_1 initial
4) If S_2 is final state, make S_1 final
5) Remove dead state.









Using Transition Table:

a) We need to make closure of every states.

ε-NFA	0	1	ε
→ q_0	∅	q_2	q_3
q_1	q_3	∅	q_4
q_2	q_0	∅	∅
q_3	q_1	∅	∅
q_4	q_5	∅	∅
q_5	∅	q_6	∅
* q_6	∅	∅	∅

	0	1
→ q_0	q_3, q_5	q_2
q_1	q_3, q_5	∅
q_2	q_0	∅
q_3	q_1	∅
q_4	q_5	∅
q_5	∅	q_6
* q_6	∅	∅

ε-Closures:

$q_0 = \{q_0, q_2, q_4\}$

$q_1 = \{q_1, q_4\}$

$q_2 = \{q_2\}$

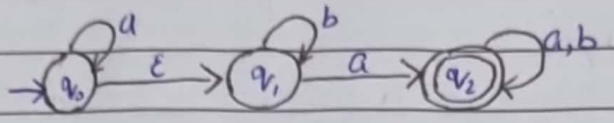
$q_3 = \{q_3\}$

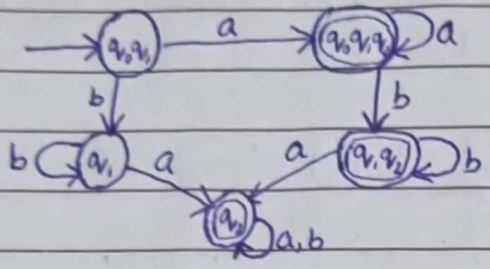
We take E closure of initial state

① E closure q_0 : $\{q_0, q_2\}$
↳ This will become our initial state of DFA

Date: _____

ε-NFA to DFA:





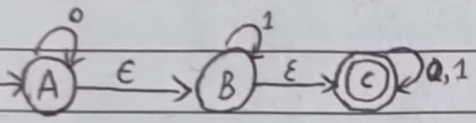
$2^n \Rightarrow 2^3 \Rightarrow 8$
max Num of states in DFA

②

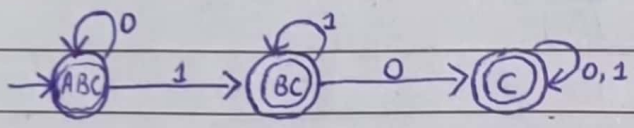
$q_0q_1 \xrightarrow{a} a(q_0) \cup a(q_1)$
↳ $q_0q_1q_2 \cup q_2$
↳ $q_0q_1q_2$

$q_0q_1 \xrightarrow{b} b(q_0) \cup b(q_1)$
$q_1 \cup q_1$
↳ q_1

$q_0q_1q_2 \xrightarrow{a} q_0q_1q_2$
$\xrightarrow{b} q_1 \cup q_1 \cup q_2$
↳ q_1q_2



① ε-Closure of A : $\{A, B, C\}$



②

$ABC \xrightarrow{0} 0(A) \cup 0(B) \cup 0(C)$
↳ ABC

$ABC \xrightarrow{1} 1(A) \cup 1(B) \cup 1(C)$
$BC \cup BC \quad C$

$BC \xrightarrow{0} 0(B) \cup 0(C)$
$C \cup C$

$BC \xrightarrow{1} BC$

Regular Language: Lang which are accepted by Finite Automata.

Lang: Collection of strings.

Regular Expressions: Date: _____

Q) Consider Alphabet $\Sigma = \{a, b\}$

1) All strings having a single 'b': a^*ba^*

2) // // // atleast one 'b': $(a+b)^*b(a+b)^*$

3) // // // 'bbbb' as substring: $(a+b)^*bbbb(a+b)^*$

4) // // // ends with 'ab' $(a+b)^*ab$

Regular Expressions: are an algebric way to describe exactly the regular languages.

→ If E is a regular expression, then L(E) is the language it defines.

We'll describe RE's and their languages recursively.

Examples:

•) $L((0+1)^*101(0+1)^*)$ = all strings of 0's and 1's having 101 as a substring.

•) $L((0+1)^*1(0+1)^*0(0+1)^*1(0+1)^*)$ = all strings of 0's and 1's having 101 as subseque

•) $L(1^*(1^*01^*01^*01^*)^*1^*)$ = all strings of 0's and 1's having a number of 0's that is multi

Even No. of a's & b's: $[aa + bb + (ab+ba)(aa+bb)^*(ab+ba)]^*$ → Even Even Language.

k-path inductive Case

$R_{ij}^{k} = R_{ij}^{k-1} + R_{jk}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}$

Date: _____

Q) What is the difference between the strings & the words of language?

A string is any combination of the letters of an alphabet where as the words of a language are the strings that are alway made according to certain rules define in lang.

Q) Valid / Invalid alphabet?

Any alphabet is valid if any of its letter doesnot appear in start of any other letter otherwise it is invalid.

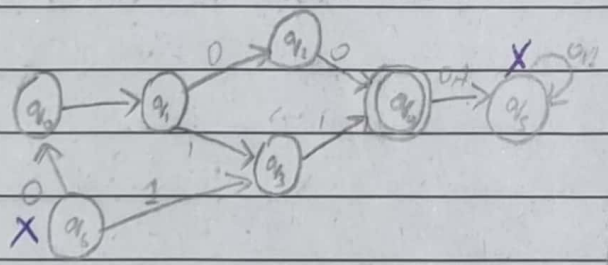
Decision Property of Regular Language: Date: _____

Language classes have two important kinds of properties (1) Decision (2) Closure
Decision Property: for a class of languages is an algorithm that takes a formal description of a language (eg: a DFA) and tells whether or not some property holds.
Eg: Is language L empty: 1) Emptiness 2) Non-Emptiness 3) Finiteness 4) Infiniteness
5) Membership 6) Equality

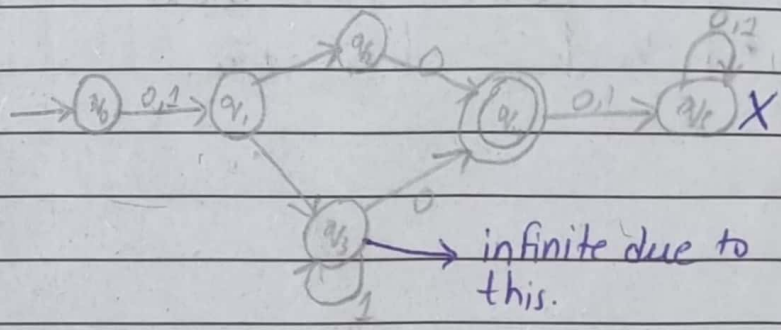
Closure Property: of language says that given languages in the class, an operator (eg union) produce another language in the same class. Example: the regular language are obv closed under union, concatenation & (kleen) closure :) Use RE representation languages.

The infiniteness Problem: Start with DFA, if the DFA has n states and the language contains any string of length n or more, then the language is infinite.
Infiniteness Test: Finding a Cycle (1) Eliminate states not reachable from the start state.
(2) Eliminate states that donot reach a final state. (3) Test if remaining transition has any cycle.

:) Remove unreachable & Deadstate.

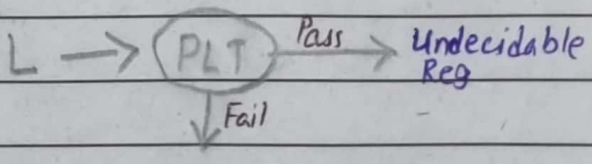


∴ its Finite



Pumping Lemma: Use to check whether our language is regular or not.
Finite lang is alway regular but infite can be regular or can't be regular.
Pumping lemma is a neg test.

(i) For each $i \geq 0$, $xy^iz \in L$
(ii) $|y| > 0$
(iii) $|xy| \leq n$



x | y | z
w

x (y) z
x yy z

...pump y, it that string is still part of lang that means 1st condition pass.

Date:

Example: $a^n b^{2n}$, $n \geq 0$

n=2 $a^2 b^{2(2)}$

$a^2 b^4$

aa bbbb $\in L$ (x y z)

aabbbb (x y z)

xy^iz

i=2, $y = (bb)^i = (bb)^2$

bbbb

aababbbb

↳ Not belong to string

∴ Non-Regular.

Now, aabbbbbb $\notin L$

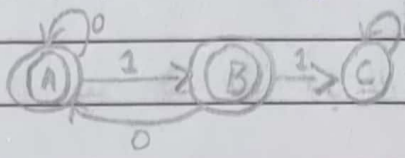
This string doesn't belong to lang.

Fail → Non-Regular.

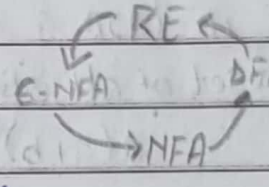
The Membership Question: 'is string w is in regular lang L'

Assume L is represented by a DFA A.

01011



if RE not represented by DF



The Emptiness Problem: Given regular lang, does the lang contain any string at all

Assume representation is DFA. Construct transition Graph.

Compute the set of states reachable from the start state

If any final state is reachable then yes, else no.

Emptiness & Non-Empty:

→ Select the state that can't be reached from the initial state & delete the unreachable states.

→ If atleast one final state is left so FA accepts the non-empty language.

→ If resulting machine is free from final state, then finite automata accepts empty language.

Infinite Union Case → RL is not closure.

Date: _____

Closure Property of Regular language: if we take 2 lang from the set of RL and perform some operation on it and output is a part of of RL then that operation is closed.

1) Union ($L_1 \cup L_2$) 2) Concatenation ($L_1 . L_2$) (3) Closure (*) L^* (4) Complementation $\bar{L} = \Sigma^* - L$
(5) Intersection $L_1 \cap L_2 = \bar{L_1} \cup \bar{L_2}$ 6) Difference $(L_1 - L_2) = L_1 \cap \bar{L_2}$ 7) Reversal $(L)^R$
(8) Homomorphism (9) Reverse Homomorphism (10) Quotiant Operation (11) INIT
(12) Substitution (13) Infinite Union.

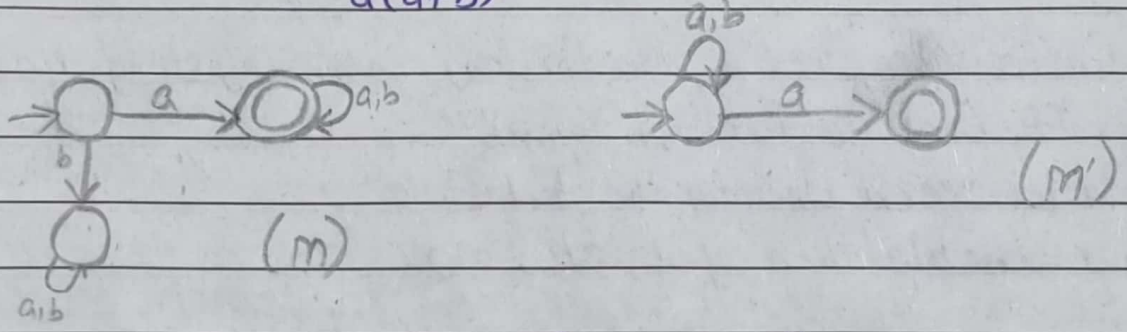
RL are not closed in infinite closure* operation otherwise its closed in all operati

RL are closed under Reversal:

1) Make the initial state of M as final state of M'.
2) Final state of M become initial state of M'.
3) Reverse the direction of edeges of M to make M'.
4) No change in loop & remove unnecessary states.

Example: L_1 = {set of all strings over (a,b) starting with a}

$a(a+b)^*$



Example: $L = \{0, 01, 100\}$ $L^R = \{0, 10, 001\}$

↳ $E = 01^* + 10^*$

$E^R = (01^* + 10^*)^R = (01^*)^R + (10^*)^R$

$= (1^*)^R 0^R + (0^*)^R 1^R$

$(1^R)^* 0 + (0^R)^* 1$

$\boxed{1^* 0 + 0^* 1}$

$E = F + G$ $E^R = F^R + G^R$

$E = FG$ $E^R = G^R F^R$

$E = F^*$ $E^R = (F^R)^*$.

Substitution Function

Date: _____

Homomorphism: on an alphabet is a function that gives a string for each symbol in that alphabet. Example: $h(0) = ab$; $h(1) = \epsilon$

$R(L) = \{h(w) \mid w \in L\}$ where $h: \Sigma$ to Γ^* is called homomorphism.

$\Sigma\{0,1\}$ $\Gamma = \{a,b\}$, $R(0) = aa$, $R(1) = bb$

if $L = \{00, 101\}$

Find $h(L) = \{aaaa, bbaabb\}$ → Homomorphic Image.

Closure Under Homomorphism: Let $h(0) = ab$; $h(1) = \epsilon$, $01^* + 10^*$

$h(L) = ab\epsilon^* + \epsilon(ab)^*$ ∴ ϵ^* can be written as ϵ

$= ab\epsilon + \epsilon(ab)^*$ ∴ ϵ represents concatenation

Thus, $ab + (ab)^*$ ∴ $L(ab)$ contained in $L(ab)^*$

So, RE for $h(L)$ is $(ab)^*$

Inverse Homomorphism $h^{-1}(L)$

Let $h(0) = a$, $h(1) = b$, $h(2) = ab$

$\Sigma = \{0,1,2\}$, $\Gamma = \{a,b\}$

Let $L = \{abab\}$

$h^{-1}(L) = \{0101, 22, 201, 012\}$

$h(h^{-1}(L)) = \{abab, abab, abab, abab\}$

$= \{abab\}$

$h(0) = aa$, $h(1) = bb$

$\Sigma = \{0,1\}$, $\Gamma = \{a,b\}$

$L = \{aa, aabb, baab, abaaba\}$

$h^{-1}(L) = \{0, 01\}$

$h(h^{-1}L) = \{aa, aabb\}$

$h(h^{-1}L) \subseteq L$

Homomorphism is always subset of Language.

$\alpha \rightarrow \beta$ ($\beta \rightarrow (V \cup T)^*$)

α → Only one variable

$G = [V, T, P, S]$

Date: _____

Grammars in theory of computation is a finite set of formal rules that are generating syntactically correct sentences. The formal defination of grammar is that it is defined as four tuples: $G = (V, T, P, S)$

G is a grammer which consist of a set of production rules. It is used to generate the string of a language.

T is the final set terminal symbols - denoted by lower case letters.

V is the final set of non-terminal symbols - Denoted by capital letters.

P is the set of production rules, which is used for replacing non-terminal symbols (on left side of production) in a string with other terminal (on right side)

S is the start symbol used to derive the string.

$V \cap T = \emptyset$

- $V = \{S, A, B\}$ → Non-Terminal (Variable)
- $T = \{a, b\}$ → Terminal Symbols
- $P = \{S \rightarrow ABa, A \rightarrow Ba, B \rightarrow ab\}$ → Production Rule (↳ Substitution)
- S $\{S\}$ → Start Symbol

A CFG is a notation for describing languages. It is more powerful than finite automata or RE's but still can't define all possible languages.

Eg: $S \rightarrow 0A$

$A \rightarrow A0 | 0$

$S \rightarrow 0S1 | \epsilon$

$0S1$

$00S11$

$000S111$

$0000\epsilon1111$

$0000 1111$

$\{0^n 1^n \quad n \geq 0\}$

$S \rightarrow 01$, $S \rightarrow 0S1$

$0S1$

$00S11$

000111

Sentential Forms: Any string of variables and/or terminals derived from the start symbol is called a sentential form.

α is sentential if $S \rightarrow \alpha$.

Variables = nonterminal → Capital letter.

Date: _____

Context-Free Languages:

A language that is defined by some CFG is called context-free language.
There are CFL's that are not regular languages.
But not all languages are CFL's.
Intuitively: CFL's can count two things, not three.

Convert CFL → CFG

1) $a^n b^n$, $n \geq 0$ — $S \to aSb \mid \epsilon$
   $a^n b^n$, $n \geq 1$ — $S \to aSb \mid ab$

2) $a^n b^{n+2}$, $n \geq 0$ — $S \to aSb \mid bb$

3) $a^{2n} b^n$, $n \geq 0$ — $S \to aaSb \mid \epsilon$
   $a^{2n} b^n$, $n \geq 1$ — $S \to aaSb \mid aab$

4) $a^{2n+3} b^n$, $n \geq 0$ — $S \to aaSb \mid aaa$

5) $a^m b^n$, $m > n$, $n \geq 0$ — $S \to AS_1$, $A \to aAa \mid a$
   $S_1 \to aS_1b \mid \epsilon$

6) $\{w \mid n_a(w) = n_b(w)\}$ — $S \to aSb \mid bSa \mid SS \mid \epsilon$
   OR $S \to aS \mid bS \mid \epsilon$

7) $ww^R \cup w(a+b)w^R$ — $S \to aSa \mid bSb \mid a \mid b \mid \epsilon$
   ↳ Palindrom Even — $S \to aSa \mid bSb \mid \epsilon$
   Odd — $S \to aSa \mid bSb \mid a \mid b$

8) $a^m b^m c^n$, $m, n \geq 0$ — $S \to S_1C$
   $S_1 \to aS_1b \mid \epsilon$
   $C \to cC \mid \epsilon$

Kleene Closure

Grammar for unsigned integer, can be replaced by:
$U \to UD \mid D$
$D \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid$

Odd Palindrome:
$S \to aSa \mid bSb \mid a \mid b$.

Even Palindrome: $|w| > 0$
$S \to aSa \mid bSb \mid aa \mid bb$

Even Palindrome:
$S \to aSa \mid bSb \mid \epsilon$.

Palindrome:
$S \to aSa \mid bSb \mid \underbrace{a \mid b}_{O} \mid \epsilon$.

Palindrome: $|w| > 0$
$S \to aSa \mid bSb \mid \underbrace{aa \mid bb}_{E} \mid \underbrace{a \mid b}_{O}$

@ Matching Parenthesis
$S \to SS \mid (S) \mid \epsilon$.

Date: _____

BNF Notation:

Grammers for programming languages are often written in BNF (Backus-Naur Form)

- Set of variables (non-terminal) and these are represend in < > like <expression>
- Set of terminal
- Production Rule [::= defined as]

Eg: $S ::= aS|SS|aSb.$

Grammer for if-then-else can be replaced by: $S \rightarrow iCtSA$

$A \rightarrow :eS|\epsilon$

CFG for Even length:

$S \rightarrow aSa|bSb|aSb|bSa|\epsilon.$

CFG for odd length:

$S \rightarrow aA|bA$

$A \rightarrow aS|bS|\epsilon$

Date: _____

Ambiguity:

Grammer may be ambiguous or unambiguous.

A grammer is said to be ambiguous grammar if it can produce either more than one left most derivation or more than one right most derivation for some string (sentence) starting from same start symbol.

$S \to AB$

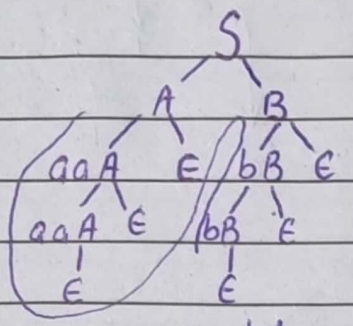
$A \to aaA \mid \epsilon$

$B \to bB \mid \epsilon$

Check whether

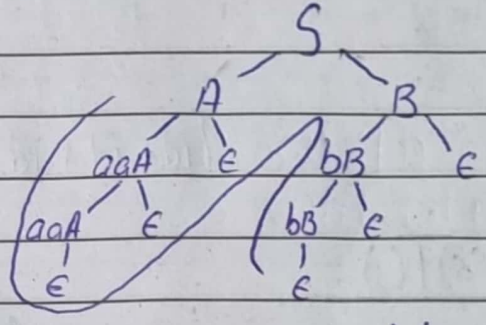
$w = aaaabb \in L(G)$

LMD Derivation tree.



aaaabb

RMD



aaaabb

Leftmost Derivation:

Balanced-parentheses grammer:

$S \to SS \mid (S) \mid ()$

$S \Rightarrow_{lm} SS \Rightarrow_{lm} (S)S \Rightarrow_{lm} (())S \Rightarrow_{lm} (())()$

Thus, $S \Rightarrow^*_{lm} (())()$

$S \Rightarrow SS \Rightarrow S() \Rightarrow (S)() \Rightarrow (())()$ is a derivation, but not a leftmost derivation.

Rightmost Derivations:

Balanced-parentheses grammer:

$S \to SS \mid (S) \mid ()$

$S \Rightarrow_{rm} SS \Rightarrow_{rm} S() \Rightarrow_{rm} (S)() \Rightarrow_{rm} (())()$

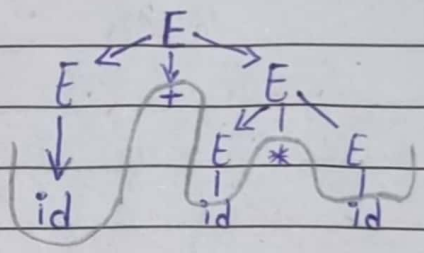
Thus, $S \Rightarrow^*_{rm} (())()$

$S \Rightarrow SS \Rightarrow SSS \Rightarrow S()S \Rightarrow ()()S \Rightarrow ()()()$ is neither a rightmost nor a leftmost derivation.

Date: _____

Parse Tree: are trees labeled by symbols of a particular CFG
Leaves: labeled by a terminal or ϵ
Interior nodes: labeled by a variable.
→ Children are labeled by the right side of a production for the parent.
Root: must be labeled by the start symbol.

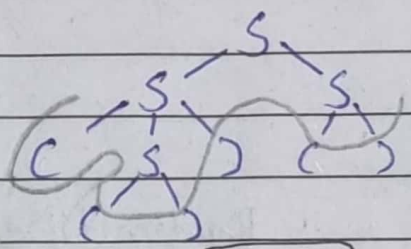
G: $E \to E+E \mid E*E \mid E=E \mid id$
W: id + id * id



id + id * id

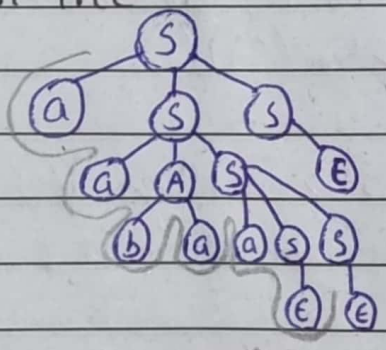
$S \to SS \mid (S) \mid ()$

(())()



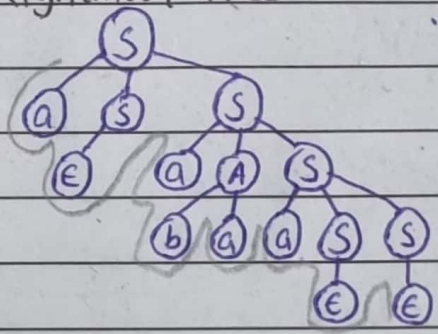
$S \to aAS \mid aSS \mid \epsilon$, $A \to SbA \mid ba$. aabaa output

Leftmost Tree



aabaa

Rightmost Tree:



aabaa

Ambiguity is a property of Grammer not language

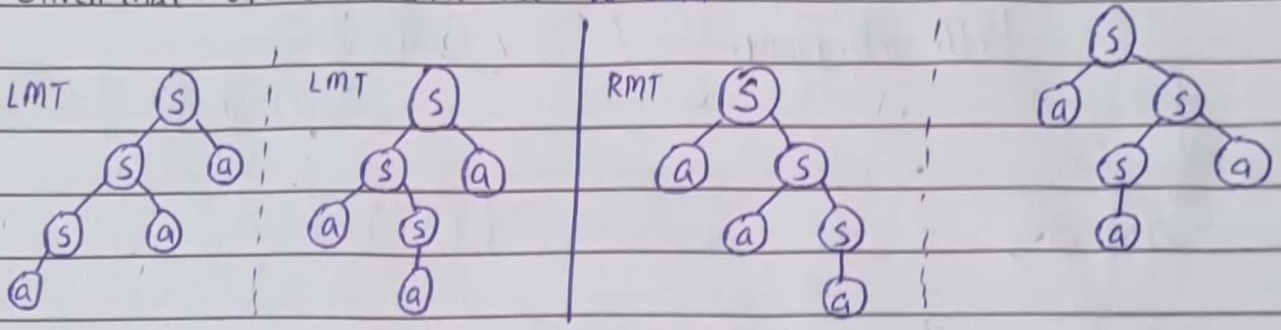
Ambiguous Grammers: Date:

A CFG is ambiguous if there is a string in the language that is the yield of two or more parse tree.

•) For ambiguous there exists more than one derivation for any word that belongs to Grammer.

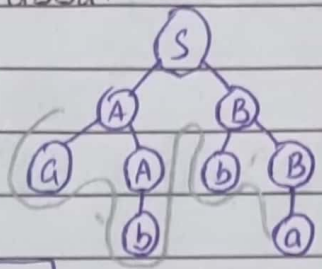
•) For unambiguous there exists exactly one derivation for any word that belongs to Grammer.

Given that: G: S → SaSaSa. W: aaa



S → AB, A → aAlb, B → bBla

W: abba.

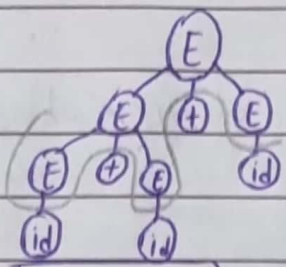


abba

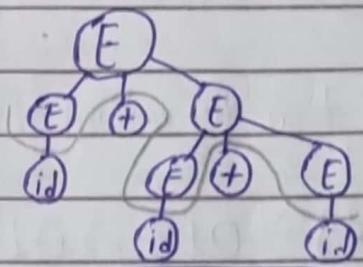
It's unambiguous as only one tree exists to make any word from this language.

E → E+E | id

W: id + id + id.



id + id + id



id + id + id

It's ambiguous as more than one tree exist for a word.

•) If lang is left & Right recursive than it's ambiguous.

E → E + E

L R

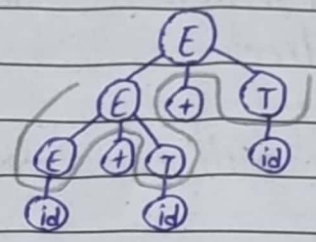
If Grammer is unambiguous LMT = RMT.

Date: _____

Conversion of Ambiguous to Unambiguous:

$E \to E+E \mid id \quad \to \quad E \to E+T \mid T$

W: id+id+id $\qquad T \to id$



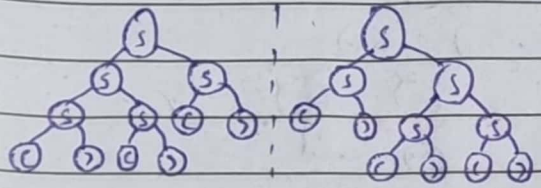
id + id + id

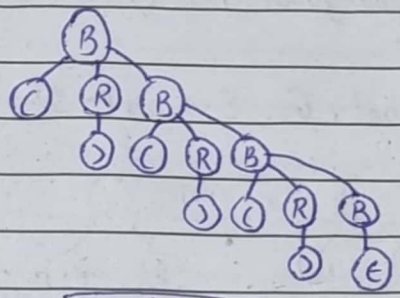
((())())

$S \to SS \mid (S) \mid ()$ $\qquad B \to (RB \mid \epsilon$

()()() ↳Ambiguous $\qquad R \to) \mid (RR$

↳Unambiguous.





(()()())

CFL $\qquad$ CFG

$0^n 1^n 2^n \to S \to AB \mid CD, A \to 0A1 \mid 01, B \to 2B \mid 2, C \to 0C \mid 0, D \to 1D2 \mid 12.$

A generates equal 0's & 1's. B generates any number of 2's. C generates any number of 0's. D generates equal 1's & 2's.

And there are two derivations of every strings with equal number of 0's, 1's & 2's.

Eg:

$S \Rightarrow AB \Rightarrow 01B \Rightarrow 012$

$S \Rightarrow CD \Rightarrow 0D \Rightarrow 012$

Recursive CFG generates infinite num of Strings

1) $S \to Sa$ {b, ba, bba, bbba, ...}
   $S \to b$
2) $S \to aS$ {b, ab, abb, abbb, ...}
   $S \to b$
3) $S \to Aa$ {ca, cba, ccba, ...}
   $A \to Ab \mid c$

Non Recursive CFG generates finite num of strings.

$S \to Aa$ {ba, ca}
$A \to b \mid c$

If LHS ∈ RHS, if any common variable then recursive

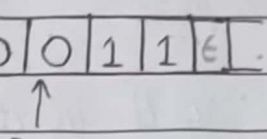
Pushdown Automata: (FA + Stack)

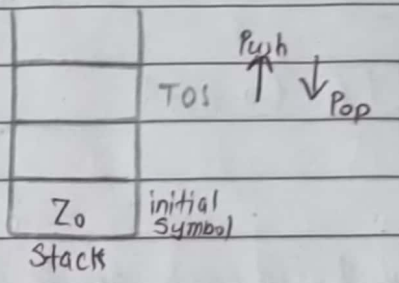
Date: _____

The PDA is an automaton equivalent to the CFG in language-defining power.
Only the nondeterministic PDA defines all the CFL's.

A PDA described by: (1) A finite set of states (Q) (2) An input alphabet (Σ) (3) A stack alphabet (Γ) (4) A transition function (δ) (5) A start state (q_0 in Q) (6) A start symbol (Z_0 in Γ). (7) A set of final states. ($F \subseteq Q$).

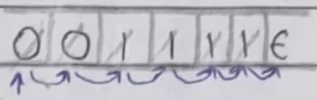
$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$



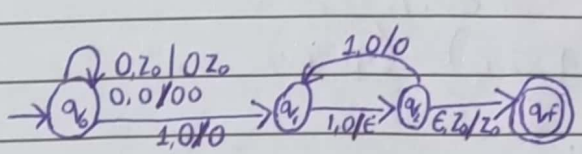


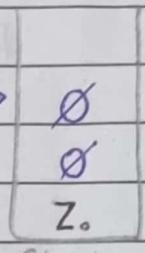
PDA Designing:

$L = \{0^n 1^{2n}, n>0\}$



Σ: 0,1 Γ: 0, Z_0





Transition:

$q_0, 0, Z_0 \rightarrow q_0, 0Z_0$

$q_0, 0, 0 \rightarrow q_0, 00$

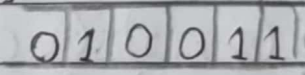
$q_0, 1, 0 \rightarrow q_1, 0$

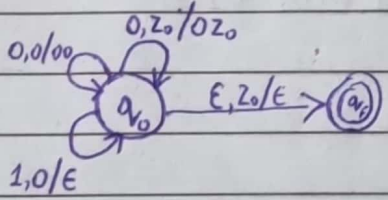
$q_1, 1, 0 \rightarrow q_2, \epsilon$

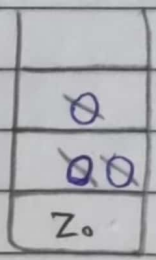
$q_2, 1, 0, \rightarrow q_1 0$

$q_2, \epsilon, Z_0 \rightarrow q_f, Z_0$

(1) $L = \{w \in (0,1)^* \mid n_0(w) = n_1(w)\}$







$\delta = Q \times (\Sigma \cup \epsilon) \times \Gamma$ to $Q \times \Gamma^*$ → Operation on stack
- Current State
- I/P
- Topmost Stack symbol
- Next State

•) Push •) Pop •) No operation.

Date: _____

PDA Example:

$(q, 000111, Z_0) \vdash (q, 00111, XZ_0) \vdash (q, 0111, XXZ_0) \vdash$
$(q, 111, XXXZ_0) \vdash (P, 11, XXZ_0) \vdash (P, 1, XZ_0) \vdash (P, \epsilon, Z_0) \vdash (f, \epsilon, Z_0)$

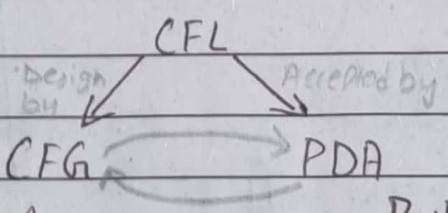
Thus, $(q, 000111, Z_0) \vdash^* (f, \epsilon, Z_0)$

Q) What would happen on input 0001111?

$(f, 1, Z_0)$

0001111 is not accepted cuz the input isn't completely consumed.

CFG to PDA



Rule 1: For each variable A

$\delta(q, \epsilon, A) = (q, B)$

where $A \to B$ is production of Grammer

Rule 2: For each Terminal 'a'

$\delta(q, a, a) = (q, \epsilon)$

$S \to 0S1 / 00 / 11$

$\delta(q, \epsilon, S) = (q, 0S1), (q, 00), (q, 11)$
$\delta(q, 0, 0) = (q, \epsilon)$
$\delta(q, 1, 1) = (q, \epsilon)$.

0111

$\delta(q, 0111, S)$ ①
$\delta(q, 0111, 0S1)$ ②
$\delta(q, 111, S1)$ ①
$\delta(q, 111, 111)$ ③
$\delta(q, 11, 11)$ ③
$\delta(q, 1, 1)$ ③
$\delta(q, \epsilon, \epsilon)$ Accepted

Date: _____

Q) $S \to OBB$ $\quad \delta(q, \epsilon, S) = (q, OBB)$ ①

$B \to OS | 1S | 0$ $\quad \delta(q, \epsilon, B) = (q, OS), (q, 1S), (q, 0)$ ②

$\delta(q, 0, 0) = (q, \epsilon)$ ③

$\delta(q, 1, 1) = (q, \epsilon)$ ④

Test: 010^4

010000

PDA to CFG

$M = \{p, q\}, \{0, 1\}, \{X, Z_0\}, \delta, q, Z_0$

Q $\quad$ Σ $\quad$ Γ

$S \to [q, Z_0, q], S \to [q, Z_0, p]$

① $\delta(q, 1, Z_0) \Rightarrow (q, XZ_0)$ Push

② $\delta(q, 1, X) \Rightarrow (q, XX)$ Push

③ $\delta(q, \epsilon, X) \Rightarrow (q, \epsilon)$ Pop

④ $\delta(q, 0, X) \Rightarrow (p, X)$ No Operation

⑤ $\delta(p, 1, X) \Rightarrow (p, \epsilon)$ Pop

⑥ $\delta(p, 0, Z_0) \Rightarrow (q, Z_0)$ No Operation

① $[q, Z_0, q] \to 1[q, X, q][q, Z_0, q]$

$[q, Z_0, q] \to 1[q, X, p][p, Z_0, q]$

$[q, Z_0, p] \to 1[q, X, q][q, Z_0, p]$

$[q, Z_0, p] \to 1[q, X, p][p, Z_0, p]$

② $[q, X, q] \to 1[q, X, q][q, X, q]$

$[q, X, q] \to 1[q, X, p][p, X, q]$

$[q, X, p] \to 1[q, X, q][q, X, p]$

$[q, X, p] \to 1[q, X, p][p, X, p]$

③ $[q, X, q] \to \epsilon$

④ $[q, X, q] \to 0[p, X, q]$

$[q, X, p] \to 0[p, X, p]$

⑤ $[p, X, p] \to 1$

⑥ $[p, Z_0, q] \to 0[q, Z_0, q]$

$[p, Z_0, p] \to 0[q, Z_0, p]$

$[q, Z_0, q] = A$, $[q, Z_0, p] = B$, $[p, Z_0, q] = C$

$[p, Z_0, p] = D$, $[p, X, q] = E$, $[q, X, p] = F$

$[p, X, q] = G$, $[p, X, p] = H$

$S \to A | B$, $A \to 1EA | 1FC$

$B \to 1EB | 1FD$, $E \to 1EE | 1FG$

$F \to 1EF | 1FH$, $E \to \epsilon$, $E \to 0G$

$F \to 0H$, $H \to 1$, $C \to 0A$, $D \to 0B$.

Date: _____

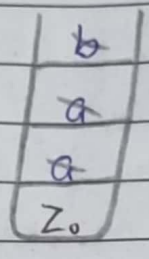
PDA For CFL

$L = \{ ww^R \mid w \in (a,b)^* \}$ 'Even Palindrome'.

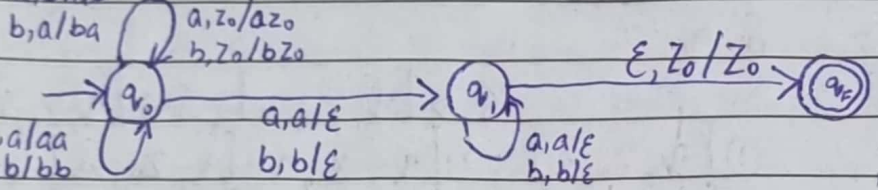
$w = aab$

$w^R = baa$

$ww^R = \overbrace{aab}^{\text{Push}}\overbrace{baa}^{\text{Pop}}$



There is no way of finding center position, therefore center position is fixed non-deterministically (NPDA).



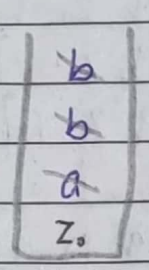
; $\delta(q_0, a, Z_0) = (q_0, aZ_0)$
; $\delta(q_0, b, Z_0) = (q_0, bZ_0)$
; $\delta(q_0, a, a) = (q_0, aa)\ (q_1, \varepsilon)$
; $\delta(q_0, b, b) = (q_0, bb)\ (q_1, \varepsilon)$] NPDA multiple moves
; $\delta(q_0, a, b) = (q_0, ab)$
; $\delta(q_0, b, a) = (q_0, ba)$
; $\delta(q_1, a, a) = (q_1, \varepsilon)$
; $\delta(q_1, b, b) = (q_1, \varepsilon)$
; $\delta(q_1, \varepsilon, Z_0) = (q_2, Z_0)$

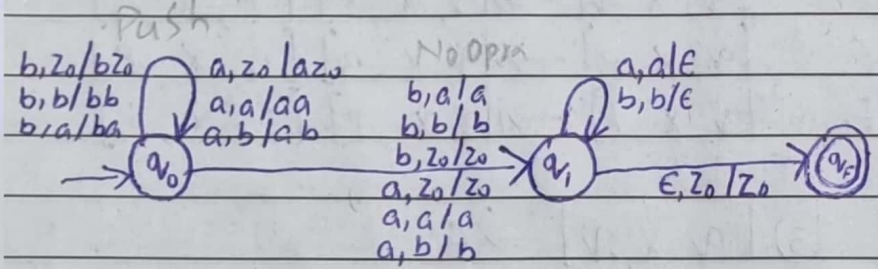
PDA For odd Palindromes NPDA

$WaW^R \mid w \in \{a,b\}^*$

WbW^R

$w: \overbrace{aba}^{w}\underset{\text{middle}}{a}\overbrace{bba}^{w^R}$





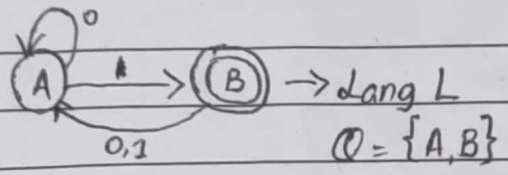
$\delta(q_0, a, Z_0) = (q_0, aZ_0), (q_0, Z_0)$
$\delta(q_0, a, a) = (q_0, aa), (q_1, a)$
$\delta(q_0, a, b) = (q_0, ab), (q_1, b)$
$\delta(q_0, b, Z_0) = (q_0, bZ_0), (q_1, Z_0)$.
$\delta(q_0, b, b) = (q_0, bb), (q_1, b)$
$\delta(q_0, b, a) = (q_0, ba), (q_1, a)$
$\delta(q_1, a, a) = (q_1, \varepsilon)$
$\delta(q_1, b, b) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, Z_0) = (q_F, Z_0)$ Final State

Decision Property: Equivalence.

Date: _____

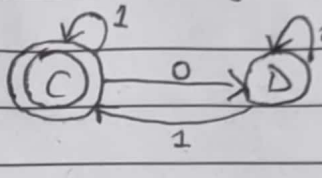
Equality: Minimize DFA, minimal DFA is unique for every RE.
If after minimization we get 2 machines save them it is equal.

Check if L = M



→ Lang L
Q = {A, B}

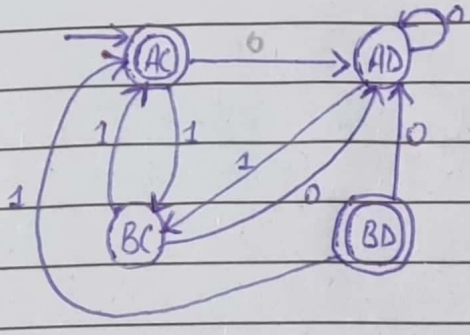
L & M are two lang.



→ lang M
R = {C, D}

Product DFA:

$Q \times R = \{A,B\} \times \{C,D\} = \{[A,C], [A,D], [B,C], [B,D]\}$ (Q, R)



	0	1
AC	AD	BC
AD	AD	BC
BC	AD	AC
BD	AD	AC

BC is not final state cuz we will get common strings but not equal strings.
To get equal strings we will take product of AC & BD then we will get equal string.

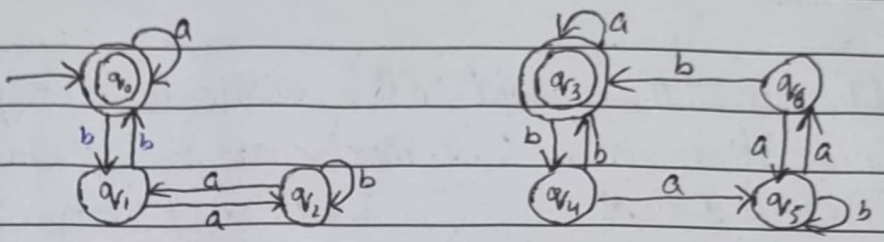
if L ∩ M the we make BC fin Sta

Containment: $L \subseteq M$

$L \subseteq M$ & L represent a empty language then what will be the final state.
BD will be final cuz it contains the final states of D. BD is empty cu no transition is towards BD.

Date: _____

Equivalence of DFA:



initial state

1) Same IS & FS in both DFA

2) $(q_0 q_3)$

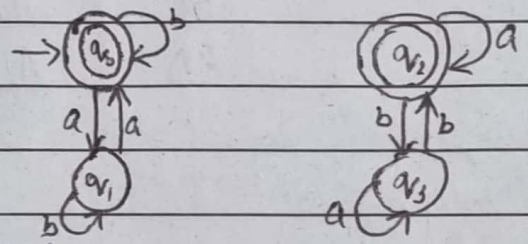
	a	b
$q_0 q_3$	$q_0 q_3$ FS FS / IS IS	$q_1 q_4$ IS IS / FS FS
$q_1 q_4$	$q_2 q_5$ IS IS / FS FS	$q_0 q_3$ FS FS / IS IS
$q_2 q_5$	$q_1 q_6$ IS IS / FS FS	$q_2 q_5$ IS IS / FS FS
$q_1 q_6$	$q_2 q_5$ IS IS / FS FS	$q_0 q_3$ FS FS / IS IS

IS = intermidiate state.

FS FS ✓
IS IS ✓
FS IS ×
IS FS ×

∴ Both are equivelant. As no conflict pair.

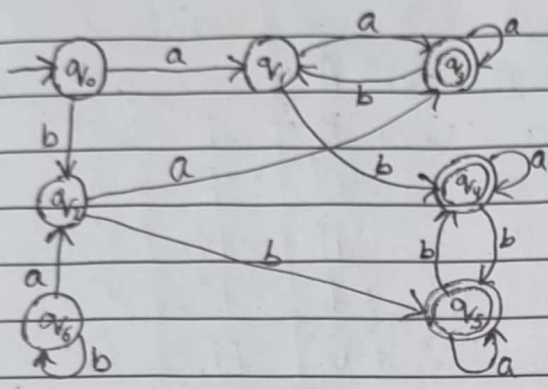
	a	b
$q_0 q_3$	$q_1 q_2$ IS FS	



Conflict pair, both aren't equivalent.

Date: _____

Minimization of DFA:



① Remove Unreachable
↳ Which is q_6

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_3	q_4
q_2	q_3	q_5
$*q_3$	q_3	q_1
$*q_4$	q_4	q_5
$*q_5$	q_5	q_4
~~q_6~~	~~q_2~~	~~q_6~~

② Classes

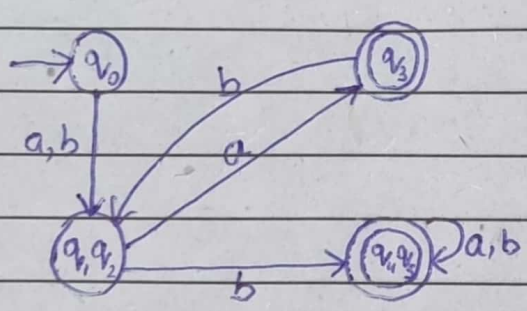
Accepting (Final) ⟷ Not Accepting (Not Final)

we need to check output of each weather they fall in same group or not. otherwise make seperate group.

$\pi_0 = \{q_0, q_1, q_2\}\ \{q_3, q_4, q_5\}$ (G_1, G_2)

$\pi_1 = \{q_0\}\ \{q_1, q_2\}\ \{q_3\}\ \{q_5, q_4\}$

$\pi_2 = \{q_0\}\ \{q_1 q_2\}\ \{q_3\}\ \{q_4 q_5\}$



Date: _____

State Minimization:

	x	b
→ A	B	C
B	D	E
C	D	F
D	D	G
E	D	G
* F	D	C
* G	D	G

$\pi_0 = \{ABCDE\}\{FG\}$

$\pi_1 = \{AB\}\{CDE\}\{F\}\{G\}$

$\pi_2 = \{A\}\{B\}\{C\}\{DE\}\{F\}\{G\}$

↓

H

	x	b
→ A	B	C
B	H	H
C	H	F
H	H	G
* F	H	C
* G	H	G

Eliminating Unreachable States.
Thus, before or after, remove states that are not reachable from start state.

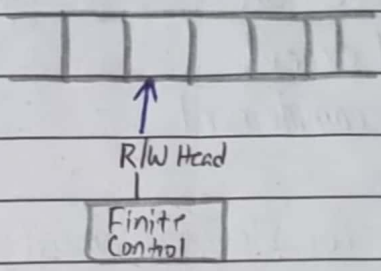
Result is the minimum state DFA

Turing Machine: Date: _____

1. A Finite set of states (Q)
2. An input alphabet (Σ)
3. A tape alphabet (Γ : contains Σ) which includes a blank symbol, B, not in Σ
   - Entire tape except for input is initially blank.
4. A transition function (δ)
5. A start state (q_0, in Q)
6. An final(accept) state (F or q_{accept})
7. A reject state (x or q_{reject}).

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

states; $\Sigma \subseteq \Gamma$

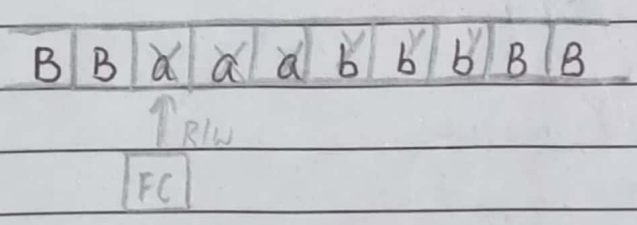


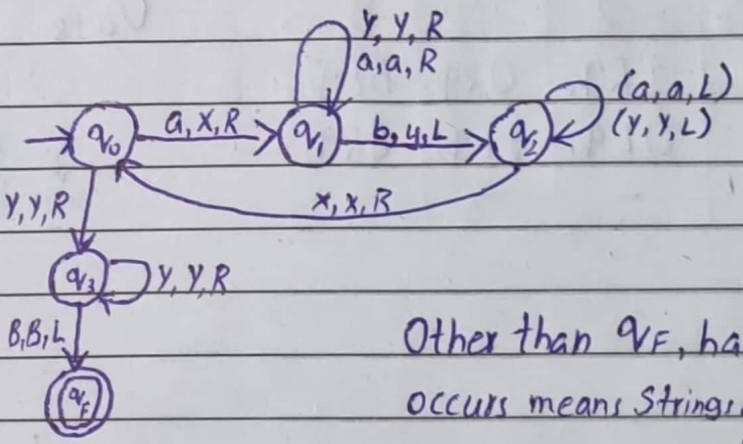
1) Read, Write
2) Left, Right.

$\delta = Q \times \Gamma \longrightarrow Q \times \Gamma \times (L, R)$

$(q_0, a) \longrightarrow (q_1, X, R)$

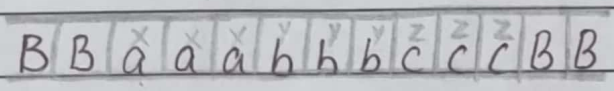
Design Turing Machine for $\{a^n b^n \mid n \geq 1\}$

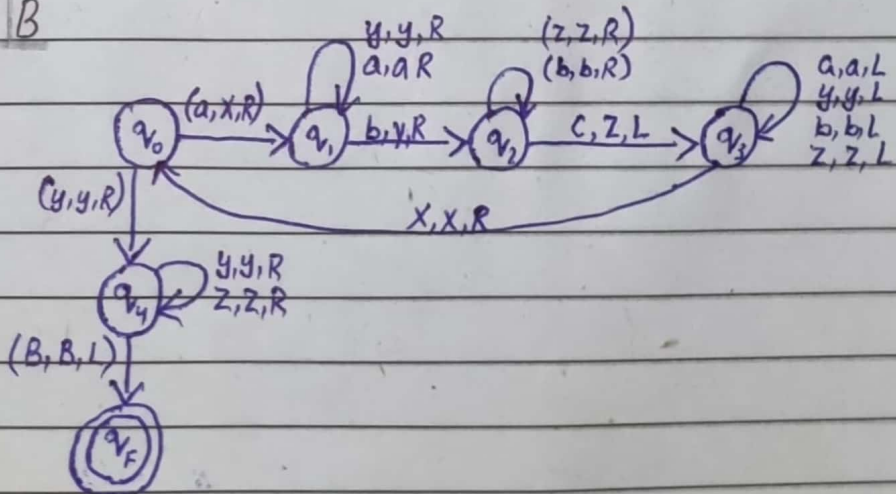




Other than q_F, halt occurs means String don't accept

TM for $\{a^n b^n c^n \mid n \geq 1\}$

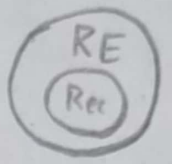




MIGHTY PAPER PRODUCT

Lang recognize by TM is Recursive Enumerable lang.

TM works as transducer.



Date: _____

Recursive Enumerable Language

→L is RE if there is TM.
→Three states: Halt & Accept
Halt & Reject
Neve Halt (loop)
→Closed under all except set difference & complement

Recursive Language:

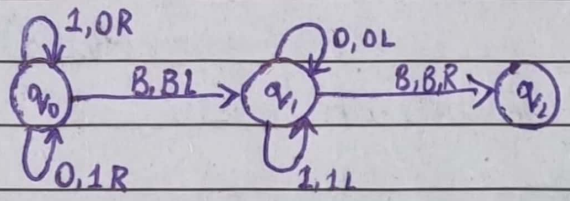
→L is Recursive if there is Halting / Total TM.
→Two States: Halt & Accept
Halt & Reject
→Closed under all except Homomorphism and substitution.

Turning Machine For 1's Complement:

1 0 1 0
0 1 0 1

-	B	B	1	1	0	0	B	B

States	0	1	B
q_0	$1Rq_0$	$0Rq_0$	BLq_1
q_1	$0Lq_1$	$1Lq_1$	BRq_2
q_2	-	-	-



α β γ → new state
↓ Change ↓ Direction

CYK → is a membership Algo.
↓
CFG

CFG
CNF GNF

Date: _____

CYK Algorithm: check wheather a string 'abbb' is valid member of following.

Valid

S → AB, A → BB|a, B → AB|b

a b b b
1 2 3 4

ab AB

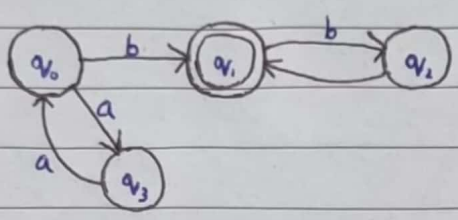
	4	3	2	1
1	S,B	A	S,B	A
2	S,B	A	B	
3	A	B		
4	B			

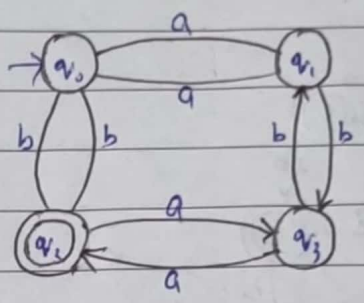
(1,4) → S is there which means its valid.

Muhammad Anas
CS221004

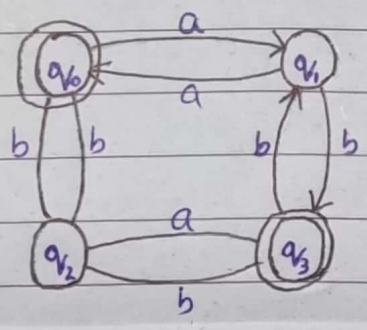
Assignment#01 Date:

Q.1)

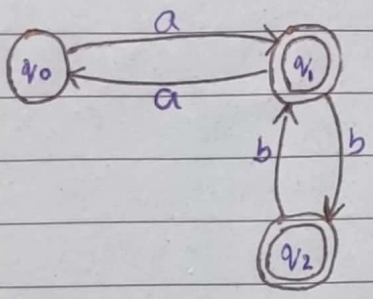


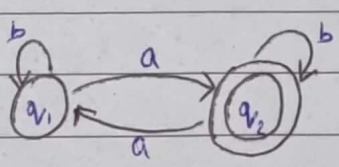


Q.2)

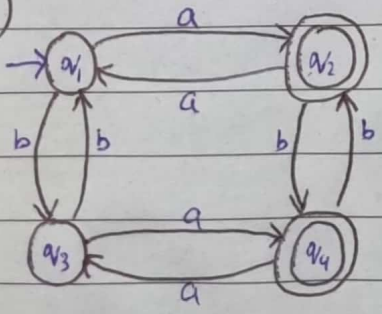


Q.3)

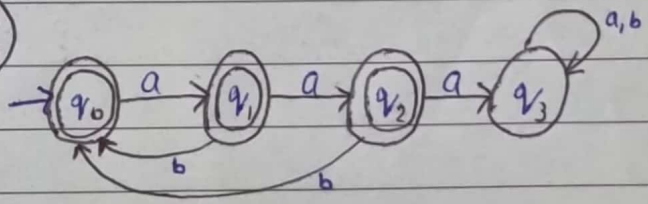




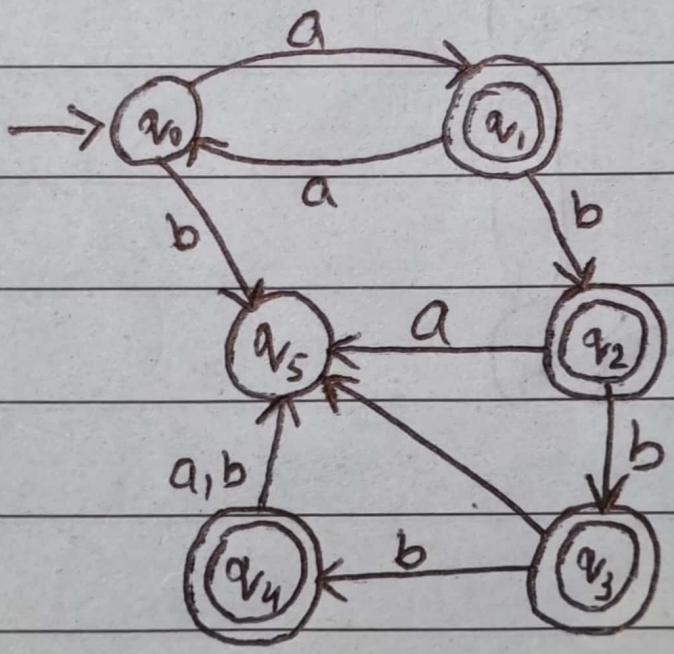
Q.4)



Q.5)



Q.4) odd a's & at most 3 b's



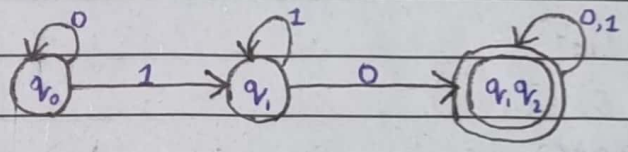
M. Anas

Assignment #02 CS221004 Date: _____

Q.2) Transition Table:

	State	0	1
→	q_0	q_0	q_1
	q_1	$q_1 q_2$	q_1
*	q_2	q_2	$q_1 q_2$

	State	0	1
→	q_0	q_0	q_1
	q_1	$q_1 q_2$	q_1
*	$q_1 q_2$	$q_1 q_2$	$q_1 q_2$



Q.1) (i) Begin & end with the same symbol.

~~$R = (x+y)(x+y)^*(x+y)$~~

$R = x(x+y)^* x + y(x+y)^* y$

(ii) Begin with an x and end with a y.

$R = x(x+y)^* y$

(iii) Have atleast one occurrence of either xx or yy

$R = (x+y)^*(xx+yy)(x+y)^*$

(iv) Consist of an even number of symbols.

$R = (xx+xy+yx+yy)^*$

(v) Contain atleast one x followed by atleast one y.

$R = (x+y)^*(xy)(x+y)^*$

CS221004 Assignment # 03 M.Anas Date: 26-6-24

(Q1) CFG for odd palindrome

$S \to pSp$

$S \to qSq$

$S \to p$

$S \to q$

Example: pSp

pqSqp

qpSpq

qpqpq → Odd Palindrome.

(Q.2)

Transition Table:

	States	0	1	B
→	q_0	$0Rq_1$	$1Rq_0$	-
	q_1	$0Rq_2$	$1Rq_0$	-
	q_2	$0Rq_2$	$1Rq_3$	-
	q_3	$0Rq_3$	$1Rq_3$	BRq_4
*	q_4	-	-	-