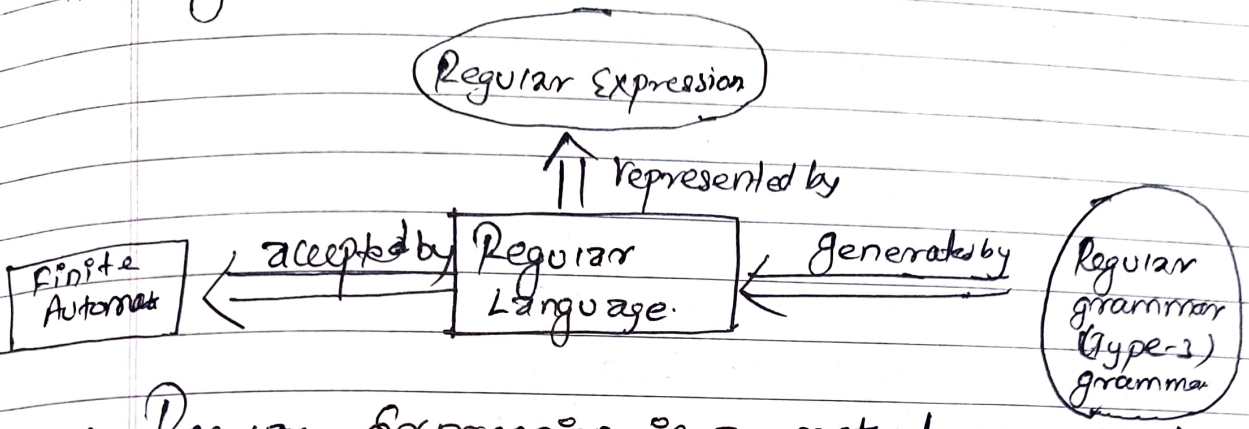
Unit-3: Regular Expressions

- Ankit Pangeni

DATE

* Regular Expressions.



→ Regular Expression is a method or way to represent the regular languages using algebric expressions.

Note: A language is said to be regular if and only if some finite state machine like DFA, NFA recognizes it.

→ Any regular Expression is composed of two components: Symbols & operators.
Symbol, Σ is a set of inputs and operators are union (∪), concatenation (.), kleen closure(*), Positive closure(+).

Ex: Regular Language L over Σ = {0,1} such that L = { w / w starts with 0 } is represented by a regular expression.

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

Note: Regular language definition also same as regular expression.
Just add: Let Σ be an alphabet, the class of regular language over Σ is defined inductively.

Formal definition of Regular Expression:

→ Let Σ be an alphabet, the regular expression over the alphabet Σ are defined inductively as:

- ϕ is a regular expression representing empty language.
- ϵ is a regular expression representing the language of empty strings. ie. the set $\{\epsilon\}$
- For each symbol $a \in \Sigma$, a is a regular expression representing the language $\{a\}$
- Union of two regular expression is also regular $(R_1 \cup R_2)$
- Concatenation of two regular expression is also regular. $(R_1 . R_2)$.
- Kleen closure * of Regular expression is also regular
- (R) is also regular. (defines parenthesis)

> For defn of regular language, in step 4, 5, 6 use regular language instead of regular expression.

Operators Of Regular Expressions

(1) Union operation:-

If L_1 and L_2 are any two regular languages, union of L_1 and L_2 denoted by $L_1 \cup L_2$ is a set of all strings that are either in L_1 or in L_2 or both.

$$L_1 \cup L_2 = \{ w / w \in L_1 \text{ or } w \in L_2 \}$$

Ex: $L_1 = \{001, 10, 111\}$, $L_2 = \{\epsilon, 001\}$

$L_1 \cup L_2 = \{\epsilon, 001, 10, 111\}$

Here, $L_1 \cup L_2$ are languages over Σ^*

Note: Closure of ϕ is finite.

DATE

② Concatenation operator :-

Concatenation of two regular languages L_1 & L_2 is the set of all strings that can be formed by taking string in L_1 and concatenating it with any string in L_2 denoted by $L_1 \cdot L_2$.

$$L_1 \cdot L_2 = \{ l_1 \cdot l_2 \mid l_1 \in L_1 \text{ and } l_2 \in L_2 \}$$

Ex: $L_1 = \{001, 10, 111\}$ $L_2 = \{\epsilon, 001\}$

$L_1 \cdot L_2 = \{001, 001001, 10, 10001, ----\}$.

③ Kleene's closure (*) operator :-

Kleene's closure of a language L is denoted by L^* and it represents the set of those strings that can be formed by taking any number of strings from L, possibly with repetitions & concatenating all of them. ie. If L is a regular language, $L^* = L_0 \cup L_1 \cup L_2 \cdots$

ie. $L^* = \bigcup_{i=0}^{\infty}$

Ex: $L_1 = \{0, 1\}$

L_1^* is the set of all strings of 0's & 1's

$L_1^* = \{\epsilon, 0, 1, 00, 01, 11, 10, 101, 0, ---\}$

Ex: $L_2 = \{0, 11\}$

$L_2^* = \{\epsilon, 011, 0110, 0111111, ---\}$

④ Positive closure (+): If L is any regular language, then positive closure of L is: $L^* = L_1 \cup L_2 \cup L_3 \cup \cdots$

ie. $L^+ = L^* - L_0$

PAGE

DATE

Special case for algebraic laws:

Note: Closure of ϕ is finite

(1) $\phi^* = \{\epsilon\}$

$\phi^0 = \{\epsilon\}$

ϕ^i for $i \geq 1$ is ϕ (empty set)

(2) $L = \{$ set of strings of 0's $\}$

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

Here,

$L^0 = \{\epsilon\}$

$L^1 = L^2 = L^3 = L^* = L$

* Application of Regular Expression

i) Validation → Determining that a string complies with a set of formatting constraints, like email validation, password validation, etc.

ii) Search & selection → Identifying a subset of items from a larger set on the basis of a pattern match.

iii) Tokenization → Converting a sequence of characters into words, tokens (like keywords, identifiers) for later interpretation.

iv) Lexer → Used as lexer/tokenizer in lexical analysis step of compilers.

classmate

PAGE

* Algebraic Laws for Regular Expression.

1) Commutativity

Union of regular expression is commutative but concatenation of regular expression is not.

Ex: If r and s are regular expression representing languages $L(r)$ and $L(s)$ then

$r+s = s+r$ ie. $r \cup s = s \cup r$

but, $r \cdot s \neq s \cdot r$.

2. Associativity:

The unions as well as concatenation of regular expressions are associative.

If l, r, s are regular expressions representing regular languages $L(l)$, $L(r)$ & $L(s)$ then

$l + (r+s) = (l+r) + s$

And $l \cdot (r \cdot s) = (l \cdot r) \cdot s$.

3. Distributive law:

~~ϕ is identity for union ie~~ for any regular expression l, m, n representing regular language $L(l)$, $L(m)$, $L(n)$ then

$l(m+n) = lm + ln$ (left distribution)

$(m+n)l = ml + nl$ (right distribution)

4. Identity Law:

ϕ is identity for union. ie. for any regular expression r representing regular expression $L(r)$

$r + \phi = \phi + r = r$ ie. $\phi \cup r = r$

ϵ is an identity for concatenation

ie. $r \cdot \epsilon = r = \epsilon \cdot r$

~~ie. $\epsilon = \epsilon$~~

5. Annihilator:

An annihilator for an operator is a value such that when the operator is applied to the annihilator & some other value, the result is the annihilator.

ϕ is annihilator for concatenation

$$\phi \cdot r = r \cdot \phi = \phi$$

6. Idempotent law of union:

For any regular expression r representing the regular language L(r),

$$r + r = r.$$

This is called idempotent law of union

7. Law of closure:

For any regular expression r representing the regular language L(r)

$$(r^*)^* = r^*$$

Closure of $\phi = \phi^* = \{\epsilon\}$

Closure of $\epsilon = \epsilon^* = \{\epsilon\}$

positive closure of $r, r^* = rr^*$

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

Note: $a^+ = aa^*$

Note: If all strings possible $(a+b)^*$

DATE

Some examples.

Q. Given, $\Sigma(a, b)$, write the regular expression for following

Finite Language

	Language	Regular Expression
1) No string	$\{\}$	ϕ
2) Length 0	$\{\epsilon\}$	ϵ
3) Length 1	$\{a, b\}$	$(a+b)$
4) Length 2	$\{aa, ab, ba, bb\}$	$aa+ab+ba+bb = (a+b)(a+b)$
5) Length 3	$\{aaa, aab, aba, \ldots, bbb\}$	$(a+b)(a+b)(a+b)$
6) At most 1	$\{\epsilon, a, b\}$	$(\epsilon+a+b)$
7) At most 2	$\{\epsilon, aa, ab, ba, bb\}$	$(\epsilon+a+b)(\epsilon+a+b)$
8) ~~Not~~ Not more than 2b's & 1a	$\{\epsilon, a, b, ab, ba, abb, bab, bba\}$	$\epsilon+a+b+ab+ba+abb+bab+bba$
9) ~~$\{\lambda, abb\}$~~	$\{\lambda, abb\}$	$\lambda + abb$
10)	$\{a, aa, aaa, aaaa, \ldots\}$	a^+
	$\{\lambda, a, aa, aaa, \ldots\}$	a^*

Q. Infinite language example.

Given $\Sigma = \{a, b\}$. Write the regular expression for following languages that contain:

a) All strings having a single 'b'

⇒ a^*ba^*

b) All strings having atleast one 'b'

⇒ $(a+b)^*\, b\, (a+b)^*$

c) All strings having bbbb as substring ⇒ $(a+b)^*bbbb(a+b)^*$

d) All strings ending with 'ab' ⇒ $(a+b)^*ab$

e) All strings start with 'ba' ⇒ $ba(a+b)^*$

f) All strings beginning & ending with a ⇒ $a(a+b)^*a$

classmate

PAGE

Note: $\lambda = \epsilon$

DATE

g) All strings containing 'a' $\Rightarrow (a+b)^* a (a+b)^*$

h) Starting & ending with different symbol

$\Rightarrow$ (a $(a+b)^* b$ + b $(a+b)^* a$)

i) " " " with same symbol $\Rightarrow a + a(a+b)^* a + b + b(a+b)^* b$

j) All strings containing 2 b's $\Rightarrow$ $a^* b a^* b a^*$

k) All strings containing 2 consecutive b's $\Rightarrow$ ~~$a^* bb a^*$~~ $(a+b)^* bb (a+b)^*$

l) $|w| = 3 \Rightarrow (a+b)(a+b)(a+b)$

m) $|w| \geq 3 \Rightarrow (a+b)(a+b)(a+b)(a+b)^* \simeq (a+b)^3 (a+b)^*$

n) $|w| \leq 3 \Rightarrow \epsilon + (a+b) + (a+b)(a+b) + (a+b)(a+b)(a+b) \simeq (\epsilon + a + b)(\epsilon + a + b)(\epsilon + a + b)$

o) Starts & ends with a $\Rightarrow a + a(a+b)^* a$

8. Give the regular expressions for following language over alphabet $\{a, b\}$

a) Set of all strings with substring bab or abb

$\Rightarrow (a+b)^* bab (a+b)^* + (a+b)^* abb (a+b)^*$

b) Set of all strings whose 3rd symbol is 'a' and 5th symbol is 'b'

$\Rightarrow (a+b)(a+b)$ a $(a+b)$ b $(a+b)^*$

c) Set of all strings containing even number of a's

$\Rightarrow$ ~~$b^* (a b^* a b^*)^*$~~ $b^* (b^* a b^* a b^*)^*$

d) Set of all strings of even length

$\Rightarrow ((a+b)^* (a+b)^*)$

e) Set of all strings ending with aa

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

$\Rightarrow$

PAGE

Note: $(a+b)^0 = \epsilon$

$(a+b)^1 = \{a, b\}$

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

Note: $\epsilon \cdot \epsilon = \epsilon$

Note: $\phi^0 = \epsilon$

DATE

8. Write languages for following regular expressions

(1) $r = \phi$, $L(r) = \{\}$ / ϕ

(2) $r = \epsilon$, $L(r) = \{\epsilon\}$

(3) $r = a$, $L(r) = \{a\}$

(4) $r = a+b$, $L(r) = \{a, b\}$

(5) $r = a \cdot b$, $L(r) = \{ab\}$

(6) $r = a+b+c$, $L(r) = \{a, b, c\}$

(7) $r = (ab+a) \cdot b$, $L(r) = \{abb, ab\}$

(8) $r = a^+$, $L(r) = \{a, aa, aaa, \ldots\}$

(9) $r = a^*$, $L(r) = \{\epsilon, a, aa, aaa, \ldots\}$

(10) $r = (a+ba)(b+a)$, $L(r) = \{ab, aa, bab, baa\}$

(11) $r = (a+\epsilon)(b+\phi)$, $L(r) = \{ab, b\}$

(12) $r = (a+b)^2$, $L(r) = \{aa, ab, ba, bb\}$

(13) $r = (a+b)^* (a+b)$ = $(a+b)^+$ e.g. $L(r) = \{a, b, aa, bb, ab, \ldots\}$

(14) $r = (a+b)^*$ (...), $L(r) = \{$ even terms $\ldots$

(15) $r = a^* \cdot a^* = \{\epsilon, a, aa, aaa, \ldots\} \cdot \{\epsilon, a, aa, aaa, \ldots\}$

$= \{\epsilon, a, \ldots\ldots a^*\}$

(16) $r = (ab)^* = \{\epsilon, ab, abab, ababab, \ldots\}$

(17) $r = \epsilon^*$, $L(r) = \{\epsilon^0, \epsilon^1, \epsilon^2, \ldots\} = \{\epsilon, \epsilon, \epsilon, \ldots\} = \{\epsilon\}$

(18) $r = \epsilon^+$, $L(r) = \{\epsilon^1, \epsilon^2, \epsilon^3, \ldots\} = \{\epsilon\}$

(19) $r = \phi^*$, $L(r) = \{\phi^0, \phi^1, \phi^2, \ldots\} = \{\epsilon\}$

(20) $r = \phi^+$, $L(r) = \{\phi^1, \phi^2, \ldots\} = \phi = \{\}$

Note: If $\Sigma = \{a, b\}$

then $\boxed{\Sigma^* = (a+b)^*}$ both same

PAGE

classmate

babaar

DATE

Some formulas

If r is a regular Expression,

(1) $r^+ \cup r^* = r^*$

(2) $r^+ \cap r^* = r^+$

(3) $r^* \cdot r^+ = r^+$

(4) $(r^*)^* = r^*$

(5) $(r^*)^+ = \{\epsilon, a, a^2, a^3, \ldots\}^+ = \{\epsilon, a^+, aa^+, aaa^+ \ldots\} = r^*$

(6) $(r^+)^* = r^*$

(7) $((r^*)^+)^* \cdot r^+ = r^* \cdot r^+ = r^+$

Ex. Given $\Sigma = \{a, b\}$. Give the regular Expression for following languages

a) $|w|_a = 2$ (ie. Strings with no. of a's 2)

⇒ $b^* a b^* a b^*$

b) $|w|_a \geq 2$ (ie. strings with no. of a's greater or equal to 2)

⇒ $(a+b)^* a (a+b)^* a (a+b)^*$

c) $|w|_a \leq 2$

⇒ $b^* (a+\epsilon) b^* (a+\epsilon) b^*$ or $b^* + b^* a b^* + b^* a b^* a b^*$

d) 3rd symbol from left end is b

⇒ $(a+b)^2 b (a+b)^*$ ⇒ $(a+b)(a+b) b (a+b)^*$

e) 28th symbol from right is a

⇒ $(a+b)^* a (a+b)^{27}$

f) $|w| = 0 \pmod 3$ | length when divided by 3 leaves remainder 0

⇒ $[(a+b)^3]^*$

DATE

g) $|w| = 2 \pmod 3$ [i.e. length supports 2, 5, 8, 11, 14

$\Rightarrow (a+b)^2 \underbrace{[(a+b)^3]}^{*}$

represents mod 3

h) $|w|_b = 0 \pmod 2$

$\Rightarrow a^*(a^* b a^* b a^*)^*$

i) $|w|_a = 1 \pmod 3$

$\Rightarrow b^* a b^* (b^* a b^* a b^* a b^*)^*$

j) $|w|_b = 2 \pmod 3$

$\Rightarrow a^* b a^* b a^* (a^* b a^* b a^* b a^*)^*$

Some Examples from class

Q. Regular expression denoting the language.

$\{0^n 1^m \mid n, m \geq 1\}$ (At least one zero & one 1

$\Rightarrow 00^* 11^*$

Q. Set of strings containing "000" as substring

$\Rightarrow (0+1)^* 000 (0+1)^*$

Q. set of strings containing even no. of 0's

$\Rightarrow$

Q. Set of strings containing 3 0's or 2 1's

$\Rightarrow (0+1)^* 0 (0+1)^* 0 (0+1)^*$ or $(0+1)^* 1 (0+1)^* 1 (0+1)^*$

$(0+1)^*$

DATE

Q- Set of strings with 3 consecutive 0's or two 1's.

⇒ $(0+1)^*(000+11)(0+1)^*$

Q. Examples of RE:

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

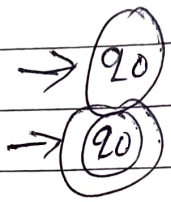
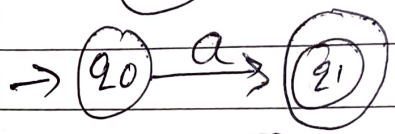
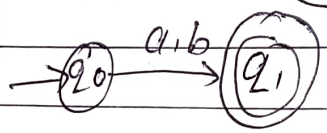
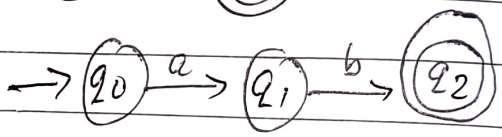
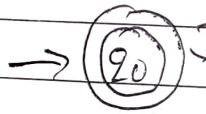
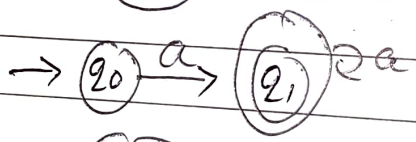
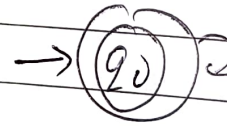
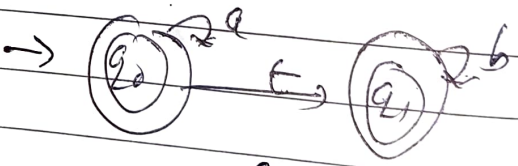
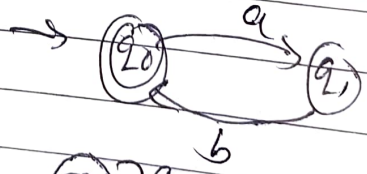
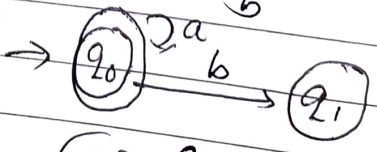
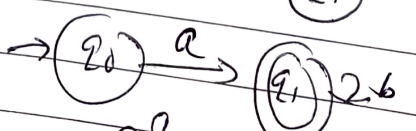
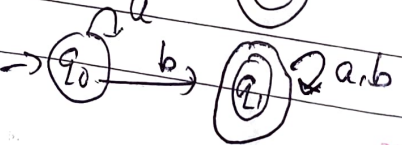
Languages	RE
L_1 = {aa, abba, ab} {length 2}	$(a+b).(a+b)$
L_1 = set of strings of length at least 2	$(a+b).(a+b).(a+b)^*$
L_2 = Set of strings of length at most 2	$(\epsilon+a+b)(\epsilon+a+b)$
L_3 = Even length strings	$((a+b).(a+b))^*$
L_4 = odd length strings.	$((a+b).(a+b))^*.(a+b)$
L_5 = set of all strings divisible by 3.	$((a+b).(a+b).(a+b))^*$
L_6 = End with aa or bb	$(a+b)^*aa + (a+b)^*bb$
L_7 = no consecutive 1's	$1+(0+01)^*$
L_8 = no consecutive 0's and 1's	$(\epsilon+1)(01)^*(\epsilon+0)$
L_9 = Even no. of 1's	$(0+(10^*1)^*)^*$

$(0+1)^*$

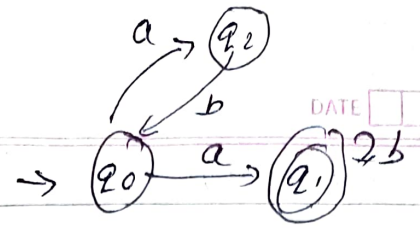
DATE

* Conversion of regular expression to Finite Automata

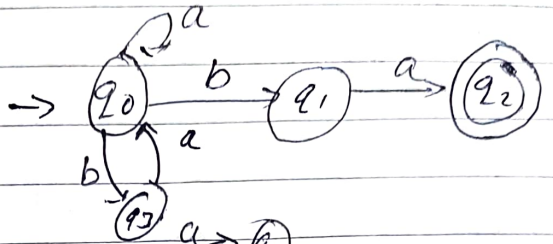
* RE → ε-NFA

REGULAR EXPRESSION	FINITE AUTOMATA
1) ϕ	
2) ϵ	
3) a	
4) a+b	
5) ab	
6) a*	
7) a+	
8) (a+b)*	
9) a*b*	
10) (ab)*	
11) a*b	
12) ab*	
13) a*b(a+b)*	

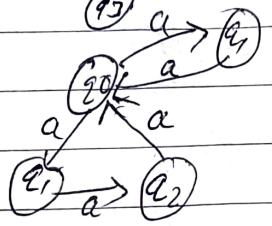
14) $(ab)^* ab^*$



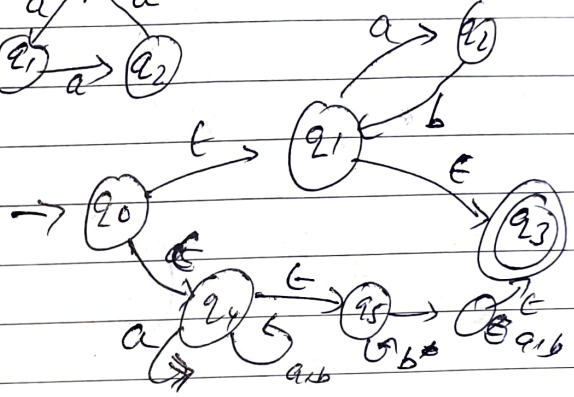
15) $(a+ba)^* ba$



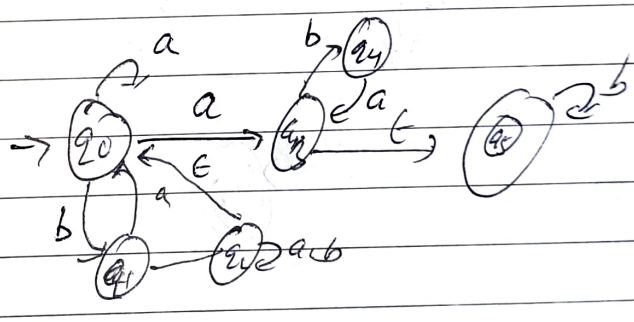
16) $(aa+aaa)^*$



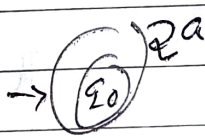
17) $(ab)^* + (a+ab)^* b^*$
$(a+b)^*$



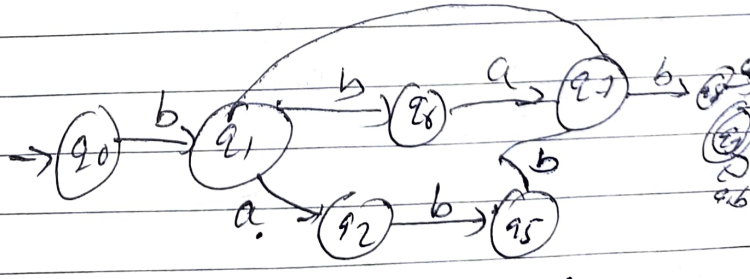
18. $[a+ba(a+b)]^*$
$a(ba)^* b^*$



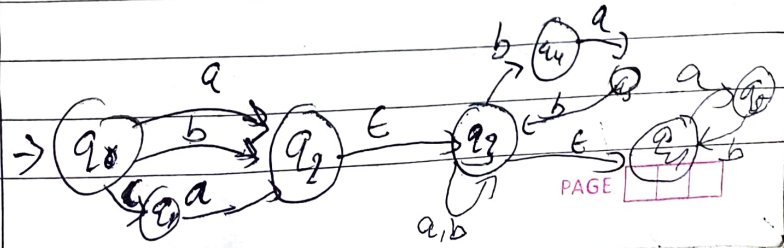
19. $(a+aaaa)^*$

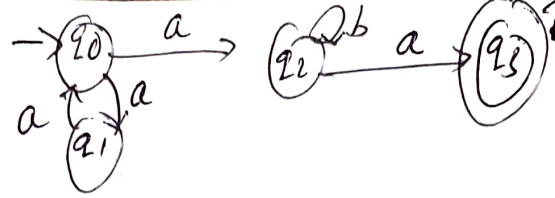


20. $b(a+ba+abb)$
$(ba(a+b)^*)$



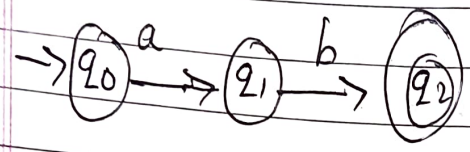
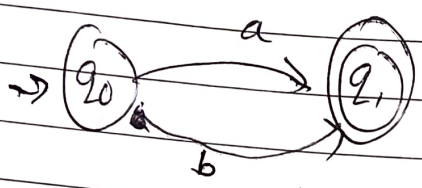
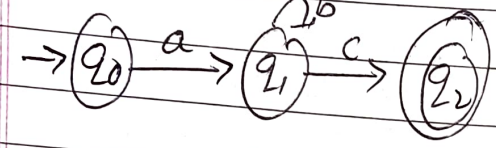
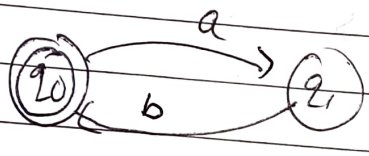
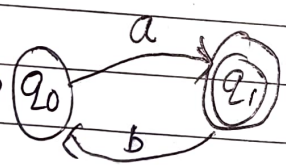
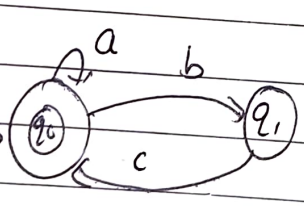
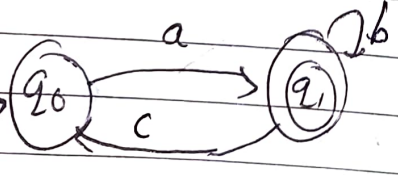
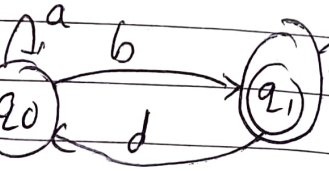
21. $(a+b+ca)((bab)^*$
$+(a+b)^*)^*(ab)^*$



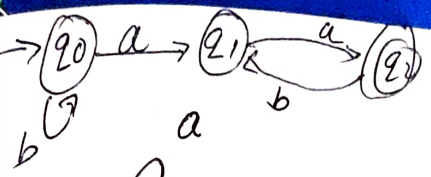
Ex:  $\Rightarrow (aa)^* a b^* a (a+b)^*$

DATE

* Conversion of FA to RE (Directly)

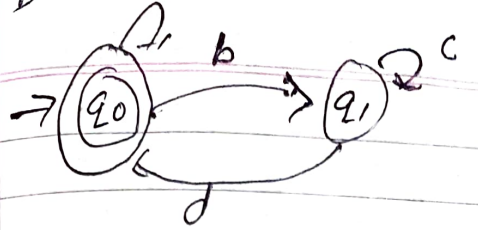
	FA	RE
1		ab
2		$a+b$
3		ab^*c
4		$(ab)^*$
5		$a(ba)^*$
6		$(a+bc)^*$
7		$a(b+ca)^*$
8		$a^*b(c+da^*b)^*$

8(i)



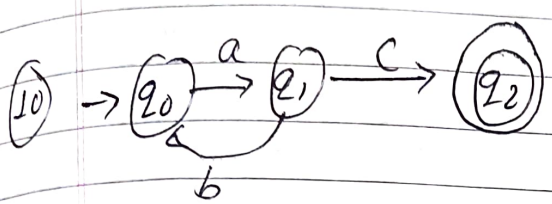
$b^* a\, a(ba)^*$

(9)



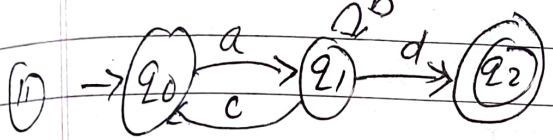
$(a + bc^*d)^*$

(10)



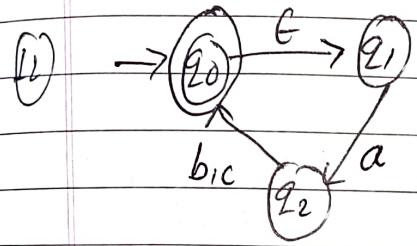
$a(ba)^*c$

(11)



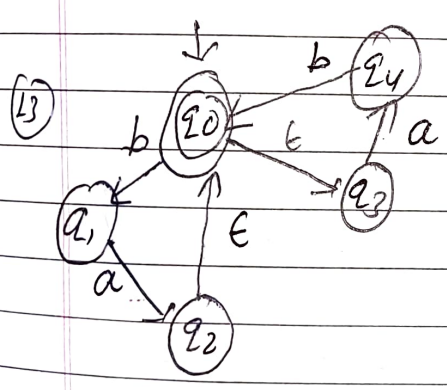
$a(b+ca)^*d$

(12)



$[\epsilon \cdot a(b+c)]^* \approx [a(b+c)]^*$

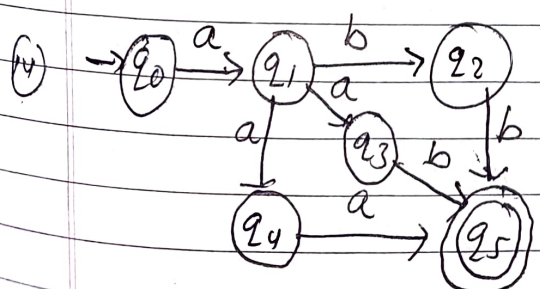
(13)



$[(ba\epsilon)^* + (\epsilon ab)^*]^*$

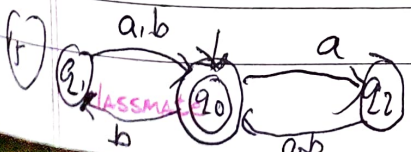
$\approx (ba + ab)^*$

(14)



$a(bb + ab + aa)$

(15)



$[a(a+b) + b(a+b)]^*$

$\approx [(a+b)^2]^*$

Closure properties of regular languages

A closure property is a statement that a certain operation on languages when applied to languages in a class (e.g the regular languages) produces a result that is also in that class.

* Informally, given a set of integers Z, performing addition operation on two integers produces an integer (eg. 4+5=9). Therefore, we say integers are closed under the arithmetic addition operation. On the other hand, performing a division operation on 2 integers may not produce an integer (eg. 5÷2=2.5). Therefore, set of integers is not closed under the division operation

* For regular Languages, we can use any of its representations to prove a closure property.

1)* Closure under union:

If L and M are regular languages, so is L∪M.

proof: Let L and M be the Languages of the regular expressions R and s respectively
Then, R+S is a regular expression whose language is L∪M
Since, a regular expression exists for L∪M, L∪M is regular ⇒ (R.L are closed under union operation)

2)* Closure under Concatenation.

If L and M are regular language, so is L.M.

Proof: Let L & M be the languages of the regular expressions R & S respectively.
Then, R.S is a regular expression whose language is L.M. Since, a regular expression exists for L.M, L.M is regular => (R.L are closed under Concatenation)

3)* Closure under star operation (Kleen's closure).

If L is a regular expression, so is L^*

Proof: Let L be the languages of regular expression R.
Then R^* is a regular expression whose language is L^*.
Since, a regular expression exists for L^*, L^* is regular => (L^* is closed under kleen closure operation)

4)* Closure under intersection:
If L & M are regular language, So is $L \cap M$.
Proof: Using cross product automation.

DATE

CLOSURE PROPERTIES OF RL

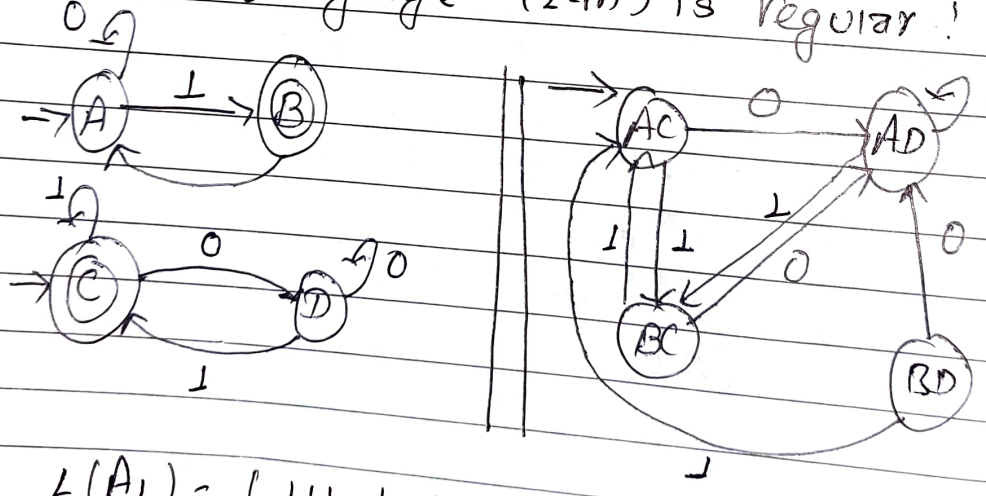
* Difference operation

-> Regular language are closed under difference operation

Proof: Let L and m be the regular languages accepted by DFA A_1 and A_2 respectively.

- Construct a cross product DFA $(A_1 \times A_2)$
- Choose final states (q_1, q_2) in the cross product DFA such that $q_1 \in A_1 \wedge q_2 \in A_2$ is a final state in A_1 and $q_2 \in A_2 \wedge q_2$ is a non-final state in A_2.

Such a cross product automation accepts since we are able to give a DFA that accepts L-m, the language (L-m) is regular!



$L(A_1) = \{ 111, 1, 00111, 01111 \ldots \}$
$L(A_2) = \{ \epsilon, 01, 1, 0011, 01111 \ldots \}$
$L(A_1) - L(A_2)$

$L(A_1)$ = Set of string ending in odd number of 1's
$L(A_2)$ = Set of string ending in at least 1

classmate

DATE

* Complementation operation :-

→ Regular Language are closed under complementation operation.

proof : The complement of a language (with respect to an alphabet Σ such that Σ^* contains L) is $\Sigma^* - L$

→ Given a regular language L over some input alphabet Σ $\bar{L}$ (ie. Complement of L) is regular

Why?

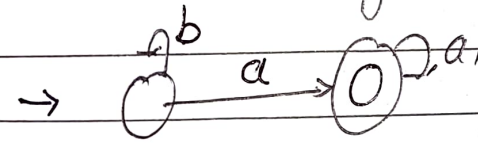
$$\bar{L} = \Sigma^* - L$$

We know Σ^* is regular and so is L. We also know that regular language are closure under difference operation

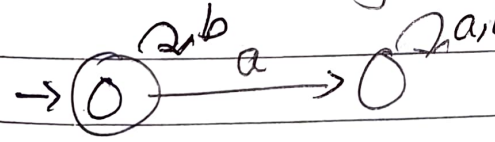
⇒ $\bar{L}$ is regular

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

L = Set of string containing 'a'



$\bar{L}$ = Set of string not containing 'a'



DATE

* Reversal Operation.

Claim: RLs are closed under reversal operation
Proof: Given a regular language L, L^R (reversal of L) is the set of strings whose reversal is in L.

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

- Let E be a regular expression for L
- We will solve how to reverse E to provide a regular expression E^R for L^R.

BASIS - If E is a symbol a, ϵ, or ϕ then $E^R = E$
INDUCTION: If E is
→ F+G, then $E^R = F^R + G^R$
→ F.G, then $E^R = G^R . F^R$
→ F^*, then $E^R = (F^R)^*$

Ex: Let, $E = 01^* + 10^*$
$E^R = (01^* + 10^*)^R$
$= (01^*)^R + (10^*)^R$
$= (1^*)^R 0^R + (0^*)^R 1^R$
$= (1^R)^* 0^R + (0^R)^* 1^R$
$= 1^*0 + 0^*1$

* Closure of RLs under Homomorphism operation

Claim: RLs are closed under homomorphism
Proof: If L is a regular language and h is a homomorphism on its alphabet

DATE

Then $h(L) = \{ h(w) \mid w \in L \}$ is a regular

- Let E be a regular expression for L
- Apply h to each symbol in E
- language of resulting RE is h(L)

Ex: Let $h(0) = ab$, $h(1) = \epsilon$

Let, L is the language of the R.E $01^* + 10^*$

Then, h(L) is the language of R.E $ab\epsilon^* + \epsilon(ab)^*$

Note: (Homomorphism)

Homomorphism on a alphabet is a function that gives a string for each symbol in that alphabet.

Ex: Given, $\Sigma = \{0, 1\}$

$h(0) = ab$, $h(1) = \epsilon$

where h is a homomorphic function

$h : \Sigma \rightarrow (a|b)^*$

This concept can be extended to strings:-

$h(c_1 \ldots\ldots c_n) = h(c_1) \ldots\ldots h(c_n)$

Ex: $h(0\,1\,0) = h(0) \cdot h(1) \cdot h(0) \cdot h(1) \cdot h(0)$

$= ab\epsilon \cdot ab \cdot \epsilon \cdot ab$

$= ababab$

PAGE

DATE

Application of closure property

(i) It helps to construct representations.
(ii) It helps to show that a language is not regular.
(iii)

Example:

use of closure property to show that the language L_2 = the set of strings with equal number of 0's and 1's is not regular.

$= \{\epsilon, 01, 10, 0101, 0011, 000111, 001011 \ldots\}$

Proof:

* we know that regular languages are closed under intersection operation
* We know that $L_1 = \{0^n 1^n \mid n \geq 0\}$ is not regular.
* we know that $L = 0^*1^*$ is regular.

If L_2 were regular, we know that

$L_2 \cap L = L_1$ must also be regular.

But, this is not the case. Therefore, L_2 cannot be regular.

DATE

Decision properties of Regular Languages

A 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

some examples of properties that are verified are:

① Membership problem

Q) Given a DFA M and a string w, is w accepted by M?

⇒ Yes. An algorithm exists to answer this question.

(1) START
(2) Scan w from left to right & make corresponding transitions in M
(3) After scanning w completely,
    If a final state in M is reached,
    M accepts w otherwise not.
(4) END

② Emptiness problem:

Q) Given a DFA M, is the language accepted by M empty?

⇒ This emptiness problem can be solved as follows.

(1) START
(2) Check if any final states of M are reachable or not
    - If yes, the language of M is not empty.
    - If no, the language of M is definitely empty.
(3) END

DATE

③ Infiniteness problem

Q. Given a DFA M, is the language accepted by M infinite?

⇒ If the DFA consists of n states and if some arbitrary string w, s.t $|w| \geq n$ is accepted by the DFA, then L(M) is infinite.

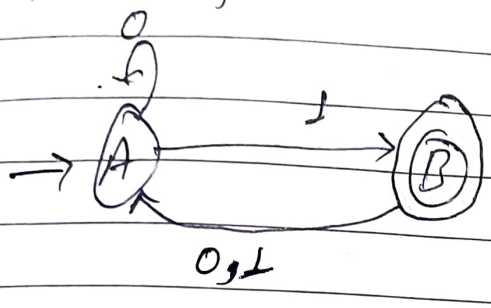
④ Equivalence Problem

Q. Given regular languages L and M, is L = M?

Soln - This question is decidable in context of regular languages. i.e. an algorithm exists that answers this question.

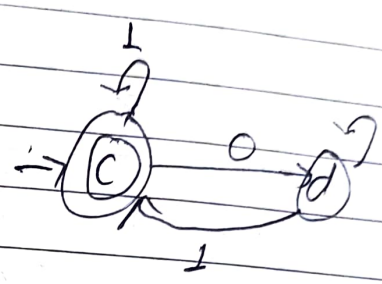
- The algorithm involves constructing the product automaton (DFA) from DFA's for L & M.
- Let these DFA's have sets of states Q & R respectively.
- Product DFA has set of states Q×R i.e. pairs [q, r] with $q \in Q$, $r \in R$.
- Make the final states of the product DFA be those states [q, r] such that exactly one of q and r is a final state of its own DFA.

- Thus, the product DFA accepts w iff w is in exactly one of L & M.
- The product DFA's language is empty iff L = M. and we already know the algorithm that decides the emptiness problem of RLs.



$L_1 = \{01, 001, 1, 1001, 111, \ldots\}$

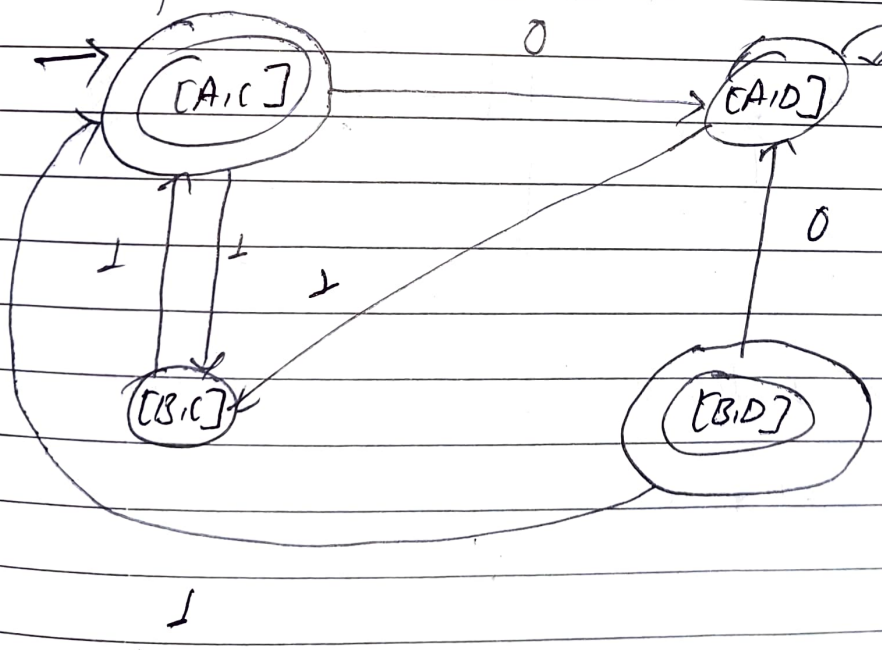
Strings ending with odd number of 1's.



$L_2 = \{\epsilon, 01, 001, 101, 1001, 0011, 111, 10111 \ldots\}$

Strings ending with 1.

Cross product



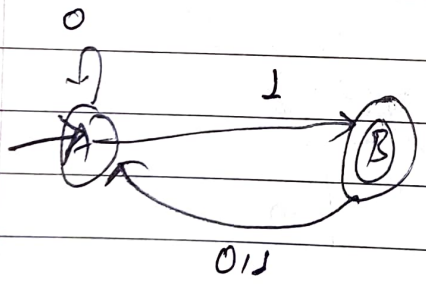
Since, language of cross product formation is not empty, $L_1 \neq L_2$.

DATE

(5) Containment Problem.

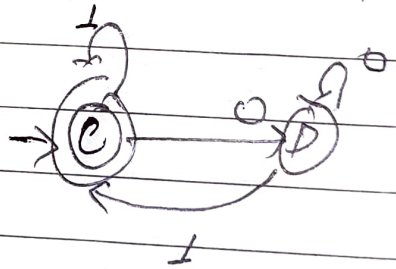
Q) Given regular language L and M is $L \subseteq M$?

- This question is decidable in context of regular languages ie. an algorithm exists that answers this question.
- The algorithm involves constructing the product (DFA) from DFA's for L & M.
- Product DFA has set of states $Q \times R$ ie pairs $[q, r]$ with $q \in Q$, $r \in R$
- Make the final states of the product DFA be those states $[q, r]$ s.t. q is final, r is non-final.
- The product DFA's language is empty iff $L \subseteq M$ and we already know the algorithm that decides the emptiness problem of RL.



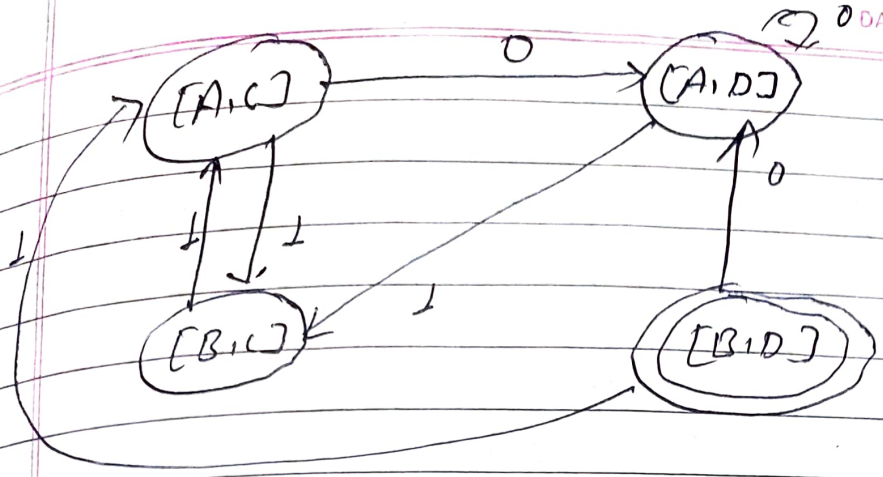
$L = \{01, 001, 1, 1001, \ldots\}$

Strings ending with odd no. of 1's.



$L = \{\epsilon, 1, 01, 001, 11, 1001, 101101 \ldots\}$

Strings ending with 1.



since, language of cross product automation is empty, so $L \subseteq M$

Pumping Lemma.

DATE

If A is a regular language, then A has a pumping length 'p' such that any string 's' where $|s| \geq p$ may be divided into 3 parts $s = xyz$ such that the following conditions must be true.

i) $xy^iz \in A$ $\forall$ $i \geq 0$

ii) $|y| > 0$

iii) $|xy| \leq p$

Pumping Lemma is used to prove that a language is not regular. It cannot be used to prove that a language is regular.

→ To prove that a language is not regular using PUMPING LEMMA, follow the steps below. (we prove using contradiction)

1) Assume that A is Regular.
2) It has to have a pumping length (say p)
3) All strings longer than p can be pumped $|s| \geq p$
4) Now find a string 's' in A such that $|s| \geq p$
5) Divide S into xyz
6) Show that $xy^iz \notin A$ for some i.
7) Then consider all ways that S can be divided into xyz
8) Show that none of these can satisfy all the 3 pumping conditions at the same time.
9) S cannot be Pumped ⇒ Contradiction

DATE

Ex: Given regular language $A = \{a^n b^n / n \geq 0\}$ Show that it is not regular

⇒ i) Suppose that A is regular

ii) Let, Pumping length = p exists such that any string $s \in A$ is $s = a^p b^p$, $|s| = 2p \geq p$

iii) Let's divide s into xyz

Assume that, $p = 7$.

Then, $s =$ aaaaaaabbbbbbb

Now, let's see all possible ways in which we can divide s into three parts xyz

Case 1: The y is in the 'a' part

aaaaaaa bbbbbbb

x y z

Case 2: The y is in the 'b' part

aaaaaaa bbbbbbb

x y z

Case 3: The y is in the 'a' as well as 'b' part

aaaaaaa bbbbbbb

x y z

iii) Now we show that $xy^iz \notin A$ for some i

xy^iz Let's take $i = 2$, then xy^2z is

Case 1: aaaaaaaaaa abbbbbbb

Since, no. of a ≠ no. of b ≠ 7, so $xy^2z \notin A$

Case 2:- aaaaaaabbbbbbbbbbbb $\notin A$

Case 3:- aaaaaaaabbaabbbbbbb $\notin A$

Also, in case 2 & case 3 $|xy| \leq p$ isn't satisfied,

Hence, it is not regular.

DATE

Ex: Given regular language $L = \{0^i 1^j \mid i > j\}$ Show that L is not regular

⇒ 1) Suppose L is regular.

2) Let pumping length $= p$ exists such that any string $S \in L$ is $S = 0^{p+1} \cdot 1^p$, $|S| = 2p+1 > p$.

3) Let's divide S into xyz

Firstly, Let pumping length $(p) = 5$

such that, $S = 0^{6} \cdot 1^5$

$\therefore S = 00000111111$

Now, let's see all possible ways in which we can divide S into three parts xyz

Case 1: The y is in '0' part

$\underbrace{0}_{x}\underbrace{00000}_{y}\underbrace{11111}_{z}$

Case 2: The y is in '1' part

$\underbrace{000000}_{x}\underbrace{11111}_{y}\underbrace{1}_{z}$

Case 3: The y is in '1' and '0' part

$\underbrace{000}_{x}\underbrace{000111}_{y}\underbrace{11}_{z}$

4) Now, we Show that $xy^iz \notin L$ for some i.

Let's take $i = 0$, then xy^0z is:

Case 1: ~~0000000000111111~~ $0 \in 11111 \notin L$

Case 2: $000000 \in L \Rightarrow 000001 \in L$

Case 3: $000 11111 \notin L$

Since, Case 1 and Case 3 Contradicts our supposition

So, the language L is not regular.

classmate

PAGE

DATE

Also, let's check other two rules $|xy| \leq p$ & $|y| \geq 0$

Here, in 1st case, $|xy| = 6 > p$
i.e. $6 > 5$
So, it is not satisfied.
So, L is not regular and hence L can't be pumped.

Ex: Given regular language $L = \{a^n b^n ; n \geq 1\}$. Show that L is not regular

a) 1) Suppose L is regular language.
2) Let pumping length = p exists such that any string $s \in L$ is $s = 0^p 1^p$, $|s| = 2p \geq p$
3) Let's divide s into xyz
First, Let's suppose $p = 3$.
Then $s = 0^3 1^3 = 000111$
Now, lets see all possible cases in which we can divide s into three parts xyz

Case 1: s is in '0' part.

"do same"

DATE

Ex: Show that $L = \{a^n \mid n \text{ is a prime number}\}$ is not a regular language.

⇒ 1) Suppose L is regular

2) Let pumping length = P exists such that any string $s \in L$ is $s = a^P$, $|s| = P \geq P$.

3) Let's divide s into xyz

First, let's suppose $P = 3$.

Then, $s = a^P = a^3 = aaa$.

Now, lets see the case in which we can divide s into three parts xyz

Case: $\underbrace{a}_{x}\ \underbrace{a}_{y}\ \underbrace{a}_{z}$.

4) Now, let's show that $xy^iz \notin L$ for some i.

Let, $i = 2$.

Then, xy^2z is $aaaa \notin L$

So, it ~~does~~ contradicts our supposition.

(Since $aaaa = a^4$ where $n = 4$ is not a prime number)

~~Also, let's check for~~

So, L is not Regular and it can't be Pumped.

DATE

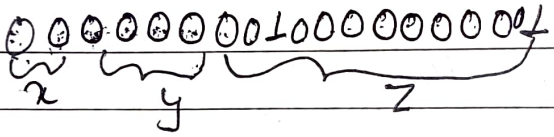
Eg: Using Pumping lemma Prove that the language $A = \{yy \mid y \in \{0,1\}^*\}$ is Not regular

1) Suppose A is regular.
2) Let pumping length = p exists such that any string $s \in A$ is.
$s = 0^p 1 0^p 1$, $|s| = 2p \geq p$.

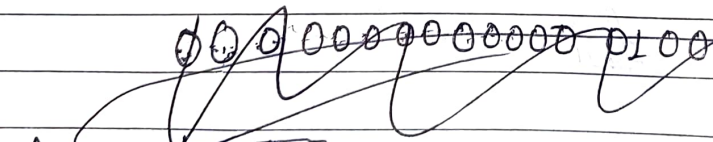
3) Let's divide s into xyz

If, let $p = 8$
Then, $s = 0^8 1 0^8 1$
$\therefore s = 000000001000000001$

Now, lets see the cases in which we can divide s into 3 parts xyz



4) Now lets show that $xy^iz \notin A$ for some i.
for, $i = 2$, xy^2z is



~~Assumption~~

$00000000000 1 000000001 \notin A$

so, it contradicts our supposition. And hence, A is not regular & it can't be pumped.

you can also check other conditions like $|xy| \leq p$ if you like

classmate

PAGE

Questions asked from this Chapter

Q. Give the regular expressions for the following language over alphabet {a,b}. (2078 - 5 marks)

a. Set of all strings with substring bab or abb.
b. Set of all strings whose 3rd symbol is 'a' and 5th symbol is 'b'.

Q. Show that $L = \{a^n \mid n \text{ is a prime number}\}$ is not a regular language. (2078 - 5 marks)

Q. Define ε-NFA & ε-closure of a state. Show that for every regular expression r, representing a language L, there is ε-NFA. (2076 - 10 marks)

Q. Give the regular expressions for following language over alphabet {0,1} (2076 - 5 marks)

a) Set of strings with 2nd symbol from right is 1.
b) Set of all strings starting with 00 or 11 and ending with 10 or 01.
c) Containing at least two 0's.

Q. Show that language $L = \{0^m 1^m \mid m \geq 1\}$ is not a regular language. (2076 - 5 marks)

Q. What is a regular grammar? Explain with language about the method of converting a regular grammar into equivalent Finite Automata (2076 (old) - 5 marks)
Convert (0+1)* 1 (0+1) to automata (2073 - 5 marks)

Q. Explain about closure properties of regular languages. Show that for any regular languages L_1 & L_2, $L_1 \cup L_2$ is also regular. (2076 (old) - 5 marks)

DATE

Q. Convert the following regular Exp to ε-NFA

a) 01^* b) $(0+1)01^*$ c) $00+(0+1)^*100^*$
d) $(00+1)^*(10)^*$ e) 001^*0^*11
f) $(0+1)^*100$ g) $(10^*+01^*)11^*$
h) $(0+1)^*(01+1000)0^*$

Q. State the pumping lemma for regular language, how can you use it to prove that a language is not regular? (2072-5 marks, 2067-5 marks)

Q. Give the regular expression for the following languages.

a) $L=\{s \mid s \in \{a,b\}^*$ and s starts with aa or b and doesn't contain sub string bb.

b) $L=\{s \mid s \in \{0,1\}^*$ and 0 occurs in pairs if any ends with 1. (2067-5 marks)