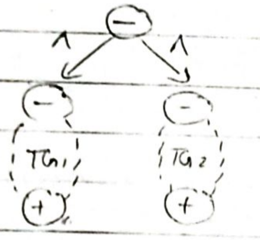
DATE: / /
PAGE NO.:

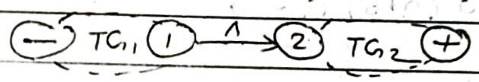
Ch.9 Regular Languages (RL)

Lang. that can be defined by a RE is called a Reg. lang.

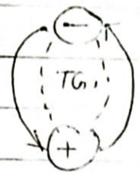
Thm: If L_1 & L_2 are RL's, then L_1+L_2, L_1L_2, L_1^* are also RL's.
i.e. the set of RL's is closed under union, concatenation & Kleene closure.

Foll. TG accept lang. L_1+L_2, TG: L_1L_2





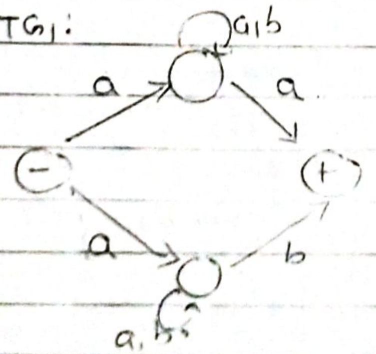
TG. L_1^*:



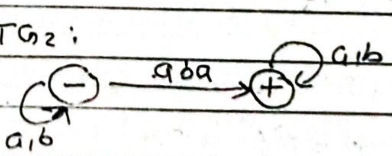
ex. Let the alphabet $\Sigma = \{a\ b\}$ &
L_1 = all words of 2 or more letters that begin & end with the same letter.
L_2 = all words that contain the substring aba.

$r_1 = a(a+b)^*a + b(a+b)^*b$ $r_2: (a+b)^*aba(a+b)^*$

TG_1:

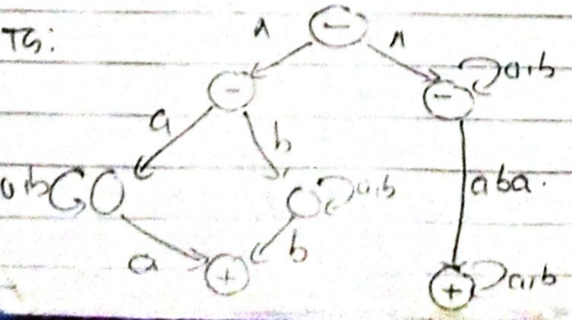


TG_2:

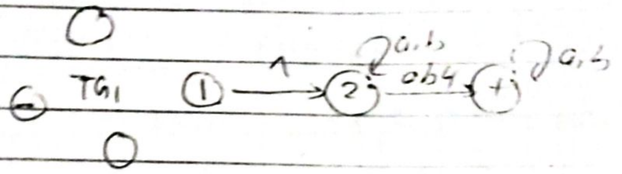


Lang. L_1+L_2 is regular because it can be defined by the RE
$[a(a+b)^*a + b(a+b)^*b] +$
$[(a+b)^*aba(a+b)^*]$.

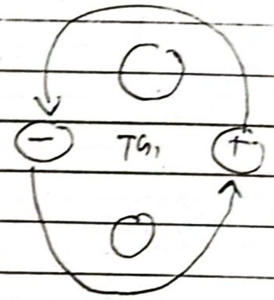
TG:



L_1L_2 is regular because it can be defined by RE
[...][...] & accepted by TG



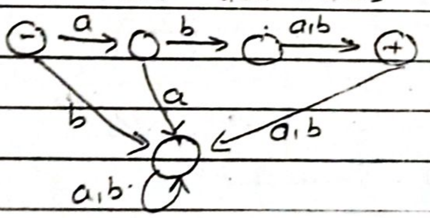
Lang. L_1^* is regular because it can be defined by RE
$[a(a+b)^*a + b(a+b)^*b]^*$
TG:



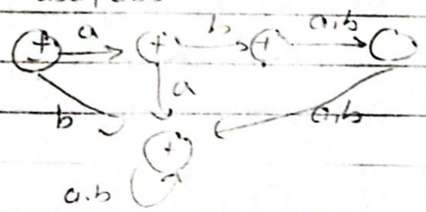
Def^n If L is a lang. over Σ, its complement L' is lang. of all strings of letters from Σ that are not words in L.

Th^m - If L is a regular lang. then L' is also RL.

ex. FA - str. aba & abb



FA accepts all str. other than aba, abb.

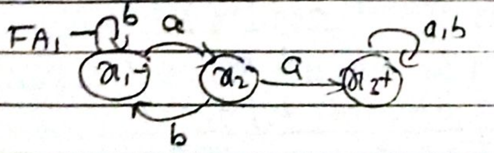


Th^m. $L_1 \cap L_2$ is also RL.
Method used here is similar to one used to produce the union-machine $FA_3 = FA_1 + FA_2$.

ex. L_1 = str. with aa

L_2 = str with even a's

FA1

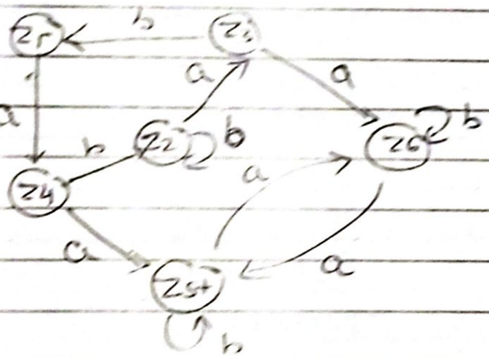


FA2 :

DATE: / /
PAGE NO:

m' that simulates the same i/p running on both m/c's at once is

	a	b	old states
$-z_1$	z_4	z_1	x_1 or y_1
z_2	z_3	z_2	x_1 or y_2
z_3	z_6	z_1	x_2 or y_1
z_4	z_5	z_2	x_2 or y_2
$+z_5$	z_6	z_5	x_3 or y_1 → final states of FA_1 & FA_2
z_6	z_5	z_6	x_3 or y_2



Nonregular languages — can't be defined by a RE.
By Kleene's th^m, it can't be accepted by FA or TG.

Pumping lemma — tool used to prove that certain lang. are nonregular.
We pump more stuff into the middle of the word, swelling it up without changing the front & back part of string.

Th^m Let L be any reg. lang that has infinitely many words. Then there exist some 3 strings x, y, z (y is not null) such that all the strings of the form,

xy^nz for $n = 1 \quad 2 \quad 3 \ldots$

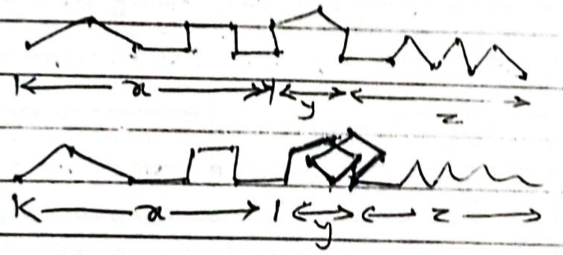
are words in L.

Proof — FA that accepts L, has only finitely many states. L has arbitrarily many words.

Let w be some word in L having more letters than No. of states in the m/c. The path for w can't visit a new state for each letter because there are more letters than states. ∴ it must at some point revisit a state.

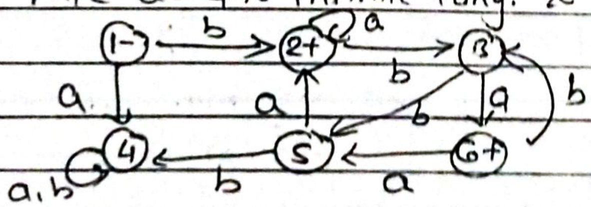
w is broken in 3 parts x — letters at beginning that lead up to the first state that is revisited. x may be null if start state is revisited first.
y — starts at letter after substring x. y is substr. that travels around the circuit coming back to same state the ckt. began with. y can't be null.
z — rest of w, can be null.
$w = xyz$, w is accepted by m/c.

if i/p is $xyyz$, path loops around ckt. twice for yy & ends in same final state. ∴ accepted.
$xyyyz, \dots$ are similar. All accepted.
∴ L contains str. of form:
xy^nz for n = 1 2 3.



Find one circuit & keep pumping it
Does not matter if there is another ckt in z.

ex. action of pumping lemma on ex. of a reg. lang.
m/c accepts infinite lang. & has only 6 states:



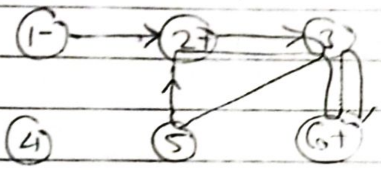
w = bbbababa — more than 6 letters ∴ ckt included.
w = b bba baba
x y z
st. 2,3,5   st. 3,6,3,6.

(1-) b (2) b (3) b (5) a (2) b (3) a (6) b (3) a (6+)
x — y — z

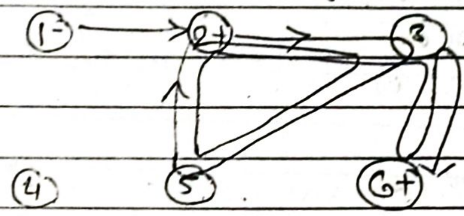
DATE: / /
PAGE NO.:

i/p = $xyyz$
= b bba bba baba.

path for xyz



path for $xyyz$



same happens with xy^nz.

ex. L = { Λ ab aabb aaabbb ...}
= { $a^n b^n$ }. for n = 0 1 2 ... }.

Apply lemma to this,
Lemma say that there must be str. x, y, z such that all words of form xy^nz are in L.
A typical word of L looks like
aaa... aaabbb... bbb

If y is made of a's, when we pump it to xyyz, word will have more a's than b's. which is not allowed in L.
Similarly for b.
If y has some positive No. of a's & b's, i.e. y contains substring ab. Then xyyz would have 2 copies of ab. But every word in L contains ab exactly once. ∴ xyyz can't be in L.

This proves that pum. lem. can't apply to L & ∴ L is not regular.

ex. Lang. Equal - { Λ ab ba aabb abab abba baab
baba bbaa aaabbb ...}.
{ $a^n b^n$ } = $a^* b^* \cap$ Equal.
If Equal were a R.L., then { $a^n b^n$ } would be $\cap$ of 2 RL's & it would have to be reg. itself. ∵ { $a^n b^n$ } is not reg., Equal can't be.

DATE: / /
PAGE NO.:

ex. lang. $a^n b a^n = \{b\ aba\ aabaa \dots\}$.

① If y contains b, then xyyz contains 2 b's, which is not allowed.

② If y is all a's, then b in the middle of xyz is in x or z side. xyyz increases no. of a's either in front of b or after b, but not both.

∴ xyyz doesn't have its b in middle & is not in form $a^n b a^n$.

This lang. can't be pumped & is ∴ not regular.