Assignment No.4 solution

Question No.1	**Pumping Lemma Version I and II**

a. Suppose we have a language defined below,

$a^n b^m$ Where m = n! (n-factorial) and n = 1, 2, 3 …

Some strings belonging to this language are,
ab , aabb , aaabbbbbb , aaaabbbbbbbbbbbbbbbbbbbbbb , …
[a^1b^1 ,a^2b^2 ,a^3b^6 ,a^4b^{24} , …]

Try to prove that this language is non-regular by Applying Pumping Lemma Version I and II on this language separately [by taking some example strings]

Solution:

Version I:
For any **Infinite Regular Language** there exists three strings x, y and z (y not null) such that all the strings of the form
xy^nz for n=1,2,3, … are the words in L.
So to prove a language as regular we need to take its one string and try to divide it in all possible ways to generate new words as shown below,

Taking string, **aabb**

x	y	z	xy^nz , n = 1,2,3…
^	aabb	^	aabb,aabbaabb,…
a	abb	^	aabb,aabbabb,…
^	aab	b	aabb,aabaabb,…
^	aa	bb	aabb,aaaabb,…
aa	bb	^	aabb,aabbbb,…
aab	b	^	aabb,aabbb,…
^	a	abb	aabb,aabbabb,…

You can see in all cases we are not able to prove that this language is regular

Version II:

Version II imposes one more restriction that for a language to be regular length of xy will be less than or equal to no. of states in FA recognizing that language.

As our language is not able to generate words using Version I so no need to apply version II on it

b. Can we say using **PUMPING LEMMA I AND II** for sure that a given language is regular or NOT.
In Pumping Lemma we are proving a language is non-regular by taking a single string from a language and applying pumping lemma both versions on it.
[A single string is enough to prove that the language is non regular and hence can NOT be regular]. So we are proving language NON REGULAR instead of proving it regular.

Question No.2	**Defining Context Free Grammars**

a. Consider the languages **EVEN** and **ODD** defined as language having strings of even and odd lengths respectively, their RE's are

Even Length Language:
((a+b)(a+b))* = (aa + ab + ba + bb)*

Odd Length Language:
((a+b)(a+b))* (a+b) = (aa + ab + ba + bb)* (a+b)

Give Context Free Grammars for these two languages separately.
Solution:

EVEN LANGUAGE CFG

S ---- > aaS|bbS|abS|baS | Є
OR
S ---- > aA|bA| Є
A ---- > aS | bS

ODD LANGUAGE CFG
S ---- > aaS|bbS|abS|baS | a | b
OR
S ---- > aA|bA| a | b
A ---- > aS | bS

b. Let us define a Language **MULTIPLE OF THREE PALINDROME** having all those strings of **PALINDROME** which have length multiple of three, some words belonging to this language are,

^ , aaa , aba , bab , bbb , aaaaaa , aabbaa , abaaba , abbbba , baaaaab , babbab , bbaabb
bbbbbb, ….
[Null string is included considering zero is also multiple of three as 0 x 3 = 0]

i. Give CFG for this language.
Solution:
S ---- > aaaSaaa | abaSaba | babSbab | bbbSbbb | aaa | aba | bab | bbb | Є

ii. Modify your CFG for **MULTIPLE OF THREE PALINDROME** for two languages below,

EVEN MULTIPLE OF THREE PALINDROME

S ---- > aaaSaaa | abaSaba | babSbab | bbbSbbb | Є

ODD MULTIPLE OF THREE PALINDROME

S ---- > aaaSaaa | abaSaba | babSbab | bbbSbbb | aaa | aba | bab | bbb

Question No.3	**Null and Nullable Transitions, Regular Context Free Grammars**

Consider the two CFG's given below,

S ---- > aAA | bBB | Є
A --- > bB | Є
B --- > aA | Є

S ---- > aAA | bBB | Є
A --- > bB | Z
B --- > aA | Є
Z--- > Є

[Here Є means null string, In CFG's we generally use Є for indicating null string instead of ^ sign]

a. Find null and null-able transitions (if any) separately in these two CFG's

Null Transitions:

S ---- > Є
A --- > Є
B --- > Є

S ---- > Є
B --- > Є
Z --- > Є

Null-able Transitions:

No one [all are already null]

A --- > Z [as z--- > Є]

b. Remove null transitions from these CFG's and give new CFG's for both cases without null transitions

CFG1

S ---- > aAA | bBB | Є

A --- > bB | Є
B --- > aA | Є

Removing A ---- > Є , B ---- > Є
S ---- > aAA | aA | a | bBB | bB | b | Є
A --- > bB | b [Due to B ---- > Є]
B --- > aA | a [Due to A ---- > Є]
Removing S ---- > Є
S ---- > aAA | bBB | aA | bB | a | b | Є

[Null production in start state can not be removed as it is implying that our language contains null string]

Complete CFG:
S ---- > aAA | bBB | aA | bB | a | b | Є
A --- > bB | b [Due to B ---- > Є]
B --- > aA | a [Due to A ---- > Є]

CFG2
S ---- > aAA | bBB | Є
A --- > bB | Z
B --- > aA | Є
Z--- > Є
Removing Z ---- > Є , A ---- > Є , B ---- > Є

Complete CFG:
S ---- > aAA | bBB | aA | a | bB | b | Є [Due to A ---- > Є , B ---- > Є]
A --- > bB | b [Due to B ---- > Є]
B --- > aA | a [Due to A ---- > Z ---- > Є]

[Null production in start state can not be removed as it is implying that our language contains null string]

Further Reading:

Finite Languages and Infinite Languages:
Finite languages
Finite languages are those which contain finite (countable) strings
Example:
Languages of all strings having length less than 3
Languages of all odd length strings having length greater than 3 and less than 10
Infinite languages
Infinite languages are those which contain infinite (uncountable) strings
Example:
Language of all strings having even length
0,2,4,6,… [infinite]
Language of all strings having length multiple of three
0,3,6,9,12… [infinite]

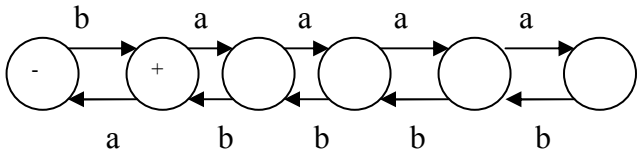
Pumping Lemma I and II:

Finite Languages:
All Finite Languages are Regular
Proof:
Simple argument to prove it is that finite languages have finite strings according to definition and we can write their RE simply by adding + operator between all these strings.
Infinite Languages:
Infinite Languages **CAN BE** regular or not.
Description:
If there is some repeated pattern in infinite language that can be checked using finite state machine (FA) than that infinite language is regular but if that infinite language doesn't contain any such repeated pattern then we can NOT make finite state machine for it.
Example:
Consider language represented by RE
(ab)*a

This language has repeated pattern of ab (two states need to recognize it irrespective of how many times it appear) next is single a one state can be used to recognize it and one dumping state will be used for all incorrect paths.

Now take any language in which there is NO SAME repeated pattern like language having a's count always ONE LESS than b's count

We can built FA for a certain length strings of this language like FA for this language for strings of length less than or equal to 9 is shown below,



But we can not build general FA for this as we don't know what length string we are going to expect and our FA CAN NOT contain infinite states. Suppose we make FA to accommodate max no. of string length say
1000000000000000000000000000000000000000000000000000000000000000
But what will happen if our string is having length,
10000000000000000000000000000000000000000000000000000000000000001
Our FA will not be able to recognize it and will produce incorrect result.
Note: There is no repeating loop in this FA that is the reason for it to have infinite states.

These concepts have been used in Pumping Lemma to find out what kind of INFINITE LANGUAGES are regular.

Pumping Lemma I and II:
In both these versions we try to find if our language FA can have any loop if so then we can simply say that our language is regular otherwise we will be in doubt and can not say anything for sure.

Since every FA has finite number of states then the language L (being infinite) accepted by F must have words of length more than the number of states, which shows that, F must contain a circuit.