Assignment Topic or Question:

Q1: Differentiate between Kleep Star Closure & Plus (+) Operator?

Kleene Star

- **Definition** − The Kleene star, $\sum^*$, is a unary operator on a set of symbols or strings, $\sum$, that gives the infinite set of all possible strings of all possible lengths over $\sum$ including **λ**.
- **Representation** − $\sum^* = \sum_0 \cup \sum_1 \cup \sum_2 \cup \ldots\ldots$ where $\sum_p$ is the set of all possible strings of length p.
- **Example** − If $\sum$ = {a, b}, $\sum^*$ = {λ, a, b, aa, ab, ba, bb,……..}

Kleene Closure / Plus

- **Definition** − The set $\sum^+$ is the infinite set of all possible strings of all possible lengths over $\sum$ excluding λ.
- **Representation** − $\sum^+ = \sum_1 \cup \sum_2 \cup \sum_3 \cup \ldots\ldots$

  $\sum^+ = \sum^* - \{ \lambda \}$
- **Example** − If $\sum$ = { a, b } , $\sum^+$ = { a, b, aa, ab, ba, bb,……..}

Q2: Define Recursive definition of factorial?

Recursive definitions

You've all seen recursive procedures in programming languages. Recursive function definitions in mathematics are basically similar.

A recursive definition defines an object in terms of smaller objects of the same type. Because this process has to end at some point, we need to include explicit definitions for the smallest objects. So a recursive definition always has two parts:

• Base case or cases

• Recursive formula/step

Recursive definitions are sometimes called inductive definitions or (especially for numerical functions) recurrence relations.

For example, recall that the factorial function n! is defined by n! = n ×

(n − 1) × . . . × 2 × 1. We can define the factorial function n! recursively:

• 0! = 1

• n! = n · (n − 1)!

The recursive definition gets rid of the annoyingly informal ". . ." and,

therefore, tends to be easier to manipulate in formal proofs or in computer

programs.

Notice that the base and inductive parts of these definitions aren't explicitly labelled. This is very common for recursive definitions. You're just

expected to figure out those labels for yourself.

Here's another recursive definition of a familiar function:

• g(1) = 1

• g(n) = g(n − 1) + n

This is just another way of defining the summation Pn

i=1 n. This particular recursive function happens to have a nice closed form:

n(n+1)

2

. Some

recursively-defined functions have a nice closed form and some don't, and it's

hard to tell which by casual inspection of the recursive definition.

Notice that both the base case and the inductive equation must be present

to have a complete definition. For example, if we leave off the base case in

the definition of g, there are quite a lot of different functions that would

satisfy the inductive condition.

Q3: Define Recursive definition of PALINDROME?

Rule 1 x POWERS

Rule 2 If a e POWERS, then ax e POWERS.

Rule 3 No other expressions are in POWERS.

Definition Polynomials in x

Rule 1 Any number is in POLYNOMIALX

Rule 2 x e POLYNOMIALX

Rule 3 If p, q e POLYNOMIALX then so are p + q, (q), and pq.

Rule 4 No other expressions are in POLYNOMIALX.

Definition Polynomials in x and y

Rule 2 becomes x, y e POLYNOMIALXY.

(i) Tell why the following recursive definition for PALINDROME over = {a, b} is incorrect, and tell how to fix it.

Rule 1 a, b e PALINDROME.

Rule 2 If x e PALINDROME, then so are axa and bxb.

Rule 3 No other strings are in PALINDROME.

(ii) Give a recursive definition for EVENPALINDROME, the set of palindromes of even length.

Solution

(i) The strings are all of odd length. Add the strings of even length.

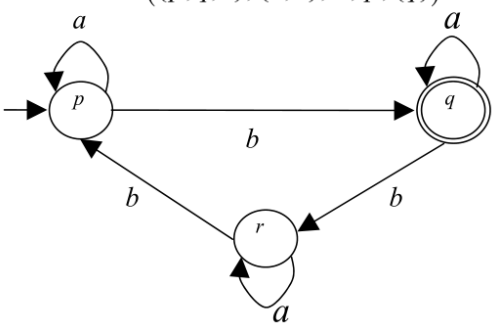
To do this, add the shortest possible string of even length, namely ⁄, in Rule 1.

(ii) Change Rule 1 to read

Rule 1 ⁄ e EVENPALINDROME.

Q4: 1) Construct a regular expression for the language accepted by the DFA using state-elimination method

Consider the DFA $A = (\{p, q, r\}, \{a, b\}, \delta, p, \{q\})$



Solution-

Step-01:

Form a equation for each state-

- q1 = ∈ ……(1)
- q2 = q1.a ……(2)
- q3 = q1.b + q2.a + q3.a …….(3)

Step-02:

Bring final state in the form R = Q + RP.

Using (1) in (2), we get-

q2 = ∈.a

q2 = a …….(4)

Using (1) and (4) in (3), we get-

q3 = q1.b + q2.a + q3.a

q3 = ∈.b + a.a + q3.a

q3 = (b + a.a) + q3.a …….(5)

Using Arden's Theorem in (5), we get-

q3 = (b + a.a)a*

Thus, Regular Expression for the given DFA = (b + aa)a*

Q5 Write regular expressions & DFA (None of your DFAs may contain more than 4 states.) for the following languages over the alphabet $\Sigma = \{a,b\}$:

(a) All strings that do not end with aa.

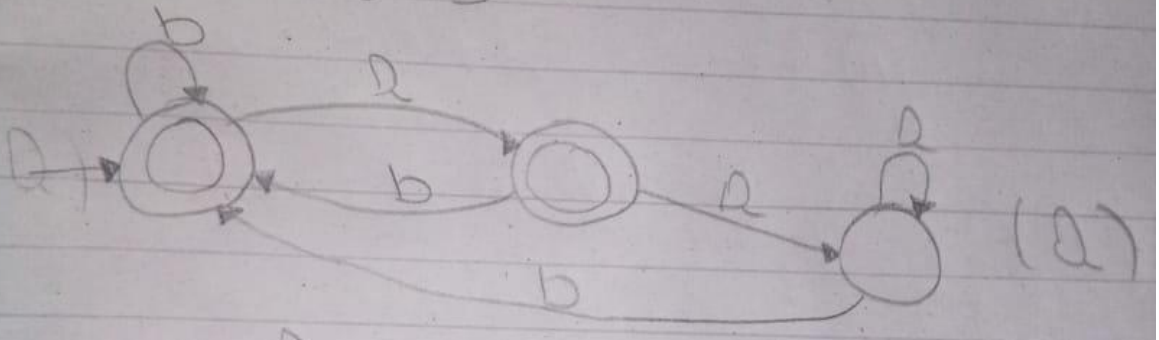
(b) All strings that contain an even number of b's.

(c) All strings which do not contain the substring ba.

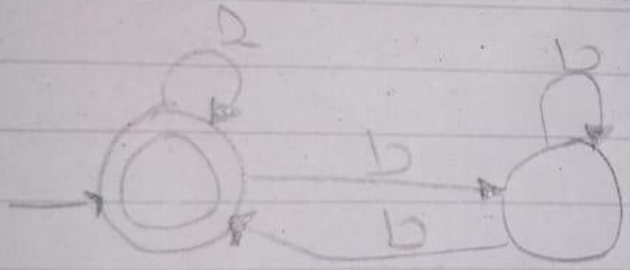
A $\varepsilon + a + b + (a+b)^*(ab+ba+bb)$

B $a^*(ba^*ba)^*$

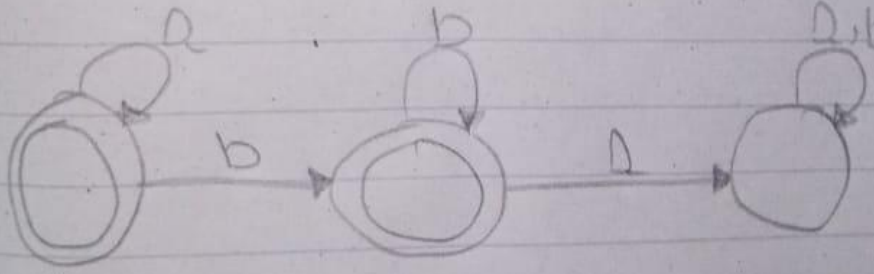
C a^*b^*



(A)



(B)



(C)