1. Show that all binary strings generated by the grammar with the following productions have values divisible by 3:

S-> 11 | 1001 | S0 | SS

The first 2 rules generate 11 which is 3 in base 2 and 1001 which is 9 in base 2. Both of which are divisible by 3. (I)

The third rule simply appends a 0 to a sequence representing a binary number. This is called shifting to the left and the effect is multiplying the number by 2 because all powers of 2 are shifted one bit to the left. This means that we add 1 to each power of 2 corresponding to a 1 digit and the result is a number which is twice as big.

Let n=3k. 2\*n = 2\*3\*k. This means that if we multiply a number that is divisible with 3 by 2 the resulting number is also divisible by three. This means that the third rule generates numbers divisible by three, assuming the base sequence is also a number divisible by three. (II)

Now, it remains to show that by concatenating 2 binary numbers that are divisible by 3, a number that is divisible by 3 results.

Let n1 and n2 be 2 numbers in binary representation, with n1 having k1 digits and n2 having k2 digits.

By concatenating n1 and n2 we obtain a number that has k1+k2 digits out of which the first k1 digits correspond to n1’s digits and the last k2 digits correspond to n2’digits. We can regard this as shifting n1 to the left by k2 digits and adding n2. This means that n1 concatenated with n2 = n1 \* 2k2 + n2.

Assume both n1 and n2 are divisible by three => n1=3\*x1 and n2=3\*x2 it follows that

n1 concatenated with n2 = 3 \* x1 \* 2k2 + 3 \* x2 = 3\*(x1 \* 2k2 + x2) which is also divisible by three. (III)

I, II, III => the productions rule generates only numbers divisible by 3 because the first 2 rules generate directly numbers divisible by 3, the third rule only multiplies a number by 2 and the fourth rule concatenates 2 binary numbers, both of those operations preserving divisibility by 3.

2. Construct a cfg for L = {anbmck | k=n+m} = { anbmcn+m | n>=0, m>=0}

S -> A | B | C

A-> ac | aAc

B-> bc | bBc

C-> aBc | aCc

I: L is in L(G)

Base case:

n=0, m=0 then the empty string epsilon

Suppose S=> anbmcn+m

Inductive step for n and for m

=>

3.  Construct a cfg for L = {w in {a,b,c}\* | noa(w) = nob(w) = noc(w)},

where noa(w)  denotes number of symbols 'a' in the sequence 'w'

S -> A | B | C

A-> ac | aAc

B-> bc | bBc

C-> aAc| aBc | aCc

abc

aabbcc, aaabbbccc, abbcca

cccbbaaab

aabbcc

S → aSb

S → bSa

S → SS

S → e

A-> a|b

B-> b|c

C->a|c

S -> aSbSc

S-> bSaSc

S-> cSaSb

S->SS

S->e

B->abc | acb | bac | bca | cab | cba | BB

| aBbc | abBc

B=>

A -> e | aAbc | abAc | AA

S=>

A -> cX | Xc| AA

X -> ab | ba

Y -> ac | ca

Z -> bc | cb

Shape, circle

Description automatically generated

delta(q, a) = q

delta(q,b) = q

delta(q, b) = q