Comp 330 - Assignment 4

# Question 1

The given language generate any string with a equals number of a and b. The grammar can produce either , so it’s always in pair so we always have an equals number of a and b’s. Now it can also be a combination of two of those string but as both have them same number of a and b’s then its still working.

We can prove using induction the inverse.

Base case: the empty string with no a’s

Let’s assume that any string s in L with at most n a’s is generated by the grammar. The goal is to show any string with a’s in L is also generated by the grammar. We now have two case:

* W starts and ends with the same letter (a or b) so contains to more bs than as and we can split w in two part(x and y) such that both part contains exactly one more b than a. So we have so now have the same number of as and bs So it can be generated by the grammar:
* W starts and ends with different letters, (or So as as less than n a’s then we can use to generate this string.

So by induction it’s true for all

# Question 2

1. Let’s make the string a+++a using two different tree

So we have the same string a+++a but they don’t mean the same a++ + a and a + ++a

1. Grammar 1:

Grammar 2

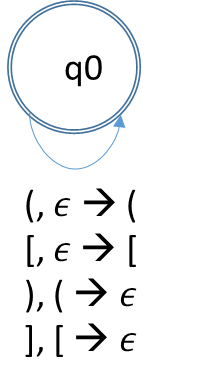
So we remove the ambiguity by putting the two different generation tree that were ambiguous in those to grammar.

# Question 3

ASB and CSD for 2., A

# Question 4

The stack can contains two element ( and [ so we can keep track of both () and []



# Question 5

We have two cases, the first check we have no less c’s than a’s and the th second that we have no less b’s than c’s. So T will do a word with the same amount of as and cs and any amount of b’s in the middle (B)

U will generate a word with the same amount of c and bs. So AU will add any amount of A’s at the beginning

Let’s prove this grammar can generate any word in L. So let now let

So we can rewrite w: . is generated by C so if is also generated by the grammar then w is too( ). Suppose we have then So we can use the rule T to generate n as and cs then the rule B tp have m bs. Simirlarty if then we can use the rule with U generating n b and c and A m as. So we can generate and using the grammar and thus w can also be generated using the grammar

# Question 6

Let find a counter example by taking the previous language where m,n,p are all strictly positive. MIN (L) is in fact the special case where . So we have

Let take

Now let , with substring of s and and . So we can easely see that with those restriction can only be composed of 2 letters. So we have the following cases:

* is only compose of a or b’s then then the amount of a’s (or b’s) is smaller than the amount of c and as c need to be the minimum it’s not working
* is only compose of c then then the amount of c is smaller than the amount of a or b
* is compose of a’s and b’s then must have more a’s and b’s than c’s
* is compose of b’s and c’s then then we have more c’s than a’s that cannot be possible as the number of c’s needs to be the min(n,m)

So all cases fail to be in so by the pumping lemma is not context free