Assignment 3

Chapter 4 and Chapter 5

50 Points

1). Show that the following grammar is ambiguous: (Please also include the two different derivation trees) (5 Points)

**S → AB | aaaB,**

**A → a | Aa,**

**B → b  
if some string within the grammar has 2 distinct derivation trees then the grammar is ambiguous. The obvious choice is aaab as that is the direct creation of making the aaaB choice for S.**A white board with red writing

Description automatically generated **Figure 1. 2 Distinct Derivation Trees for creating the string aaab.**

2). Give the derivation tree for **((a + b) ∗c+d)** using the grammar: (10 points)

**(The \* sign here is multiplication) E🡪 T, T🡪F, F🡪I, E🡪E+T, T🡪T \* F, F🡪( E ), I🡪a|b|c|d**A red arrows pointing to a plane

Description automatically generated with medium confidence

**Figure 2. Derivation Tree for((a + b) ∗c+d)**

3). Let L= **{0n 1n | n>=0}**. Using **pumping lemma**, what can be said about this language? (10 Points)  
  
Pumping lemma states that if a string not in the language can be generated from a pumped substring then the language is not regular.   
  
A red text on a white background

Description automatically generated

Figure 3. The Pumped Substring of 0000 creates a string of 00000000011111, or 0^9, 1^5. This would be a violation of the rule that n is the same for both exponents. Therefore the language is irregular.

4). Given context-free grammars that generate the following languages/ regular expressions. (15 Points)

**a) r= (ab)\*a  
This language describes any number of zero or more ‘ab’ followed by a total of one a.**A group of red arrows

Description automatically generated **Figure 4a. A CFG that generates any combination of ab followed by a.  
  
b) L = {(a, b) | b n a \* b n, n >= 0}  
  
This language describes a language composed of a and b where there can be ANY number of b’s (as long as there’s at least one) on either side of any number of a’s**A group of red letters

Description automatically generated

**Figure 4b. A CFG where zero or more ‘a’ are surrounded on either side by at least 1 b.**

**c) All strings of a and b that include the substring baa**A close-up of several arrows

Description automatically generated

**d) {w ∈ {0, 1} \*| w contains at least three 1s}  
All sets of strings of 0,1 that have a MINIMUM of THREE ONES NO MATTER THEIR POSITION.  
The solution to this in my mind is to simply have a minimum of three recursive steps, where each step must end with 1**A close-up of a number

Description automatically generated **Figure 4.d – A CFG where every combination of 0 and 1 can be generated with a minimum of three 1s.**

**e) *L* = *{anbm*: *n ≤ m* + 3*}*.**The described language is composed of ab where there must be less a’s than the number of b’s plus 3. This can be thought of as a counter variable n which is increased by one everytime a b is created. The counter variable can terminate early. The counter variable starts at 3.A group of red letters

Description automatically generated

**Figure 5e. A CFG where there are always less ‘a’ than the total number of ‘b’+3.**

5). Given the following context-free grammar, Show the derivation of the string **xyyxa** starting from S (specify which production rule you used at each step), and give the parse tree according to that derivation. (10 Points)

1. **S → Aa**
2. **A → xBC**
3. **A → CB**
4. **B → yB**
5. **B → λ**
6. **C → x**

A white board with red arrows and black letters

Description automatically generated

Figure 5. Parse tree to generate xyyxa