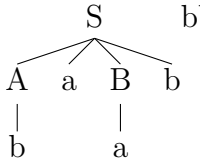
CSCI 3155 Spring 2018
Ariel Riggan and Cody Luhmann
01/28

1. Survey Completed

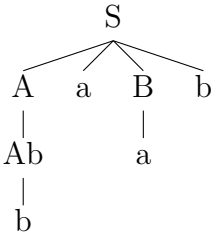
2. (a) The language defined by this grammar includes strings that begin with a number of "a"s greater than or equal to 1. Therefore your string could start with a..., aa...,aaa...., etc. The middle of strings is characterized by this grammar to have an equal number of "b" and "c" characters. There do not have to be any "b" and "c" characters in the string. The end of this language's strings must also end with at least one a, but it could be more. This is similar to the beginning of the string, but the "a"s at the beginning are independent of the number of "a"s at the end of the string.

   (b) baab, and bbaab are both strings that are defined by this grammar.

   baab:

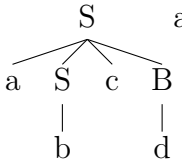


   bbaab:

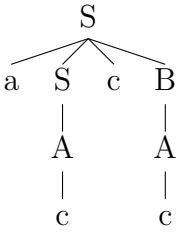


   (c) abcd and accc are both in the language defined by the grammar.

   abcd:

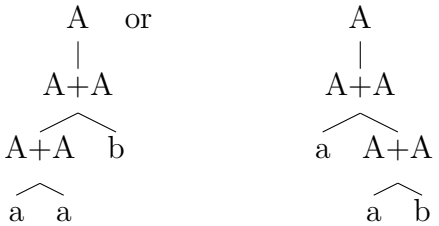


   accc:



   (d) Consider creating the parse tree for a + a + b. There are two ways with the current grammar with which this can be

done.



Being able to create the parse trees differently for the same string indicates that this grammar is ambiguous.

(e) $\frac{}{a\downarrow 1}$, $\frac{}{b\downarrow 0}$, and $\frac{x\downarrow n, y\downarrow m}{x \bigoplus y\downarrow n+m}$ These inference rules say that for the A $\bigoplus$ A, the amount of "a"s is equal to the amount of "a"s in x–the right side of the $\bigoplus$–and the "a"s in y–the left side of the $\bigoplus$. For the case where we are looking at a there is one "a" and for the case when we are looking at "b" there are zero "a"s. Using our inference rules we can go through the tree and break it down until we reach a base base, a or b and then return through our earlier problems answering the amount of "a"s for each step.

3. (a) i. The first grammar will either produce a terminal number (operand) or will do some operation with another operand on e. This could result in e, e operator operand, e operator operand operator operand, etc.
   The second grammar creates the same language as the first grammar, however it has different associativity. The first grammar is left associative and the second is right associative which just indicates that their parse trees are built differently.

   ii. These grammars generate the same expressions, but they build those expressions differently. So the final products are the same but the intermediate steps are different.

   (b)

   (c)
```
S::= -M | M
M::= NC.CENC | 0.0E0
N::=1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
C::= ZC | NC | N | Z
Z::=0
```
   Terminals: 0,1,2,3,4,5,6,7,8,9, E