Andrew Rutherford

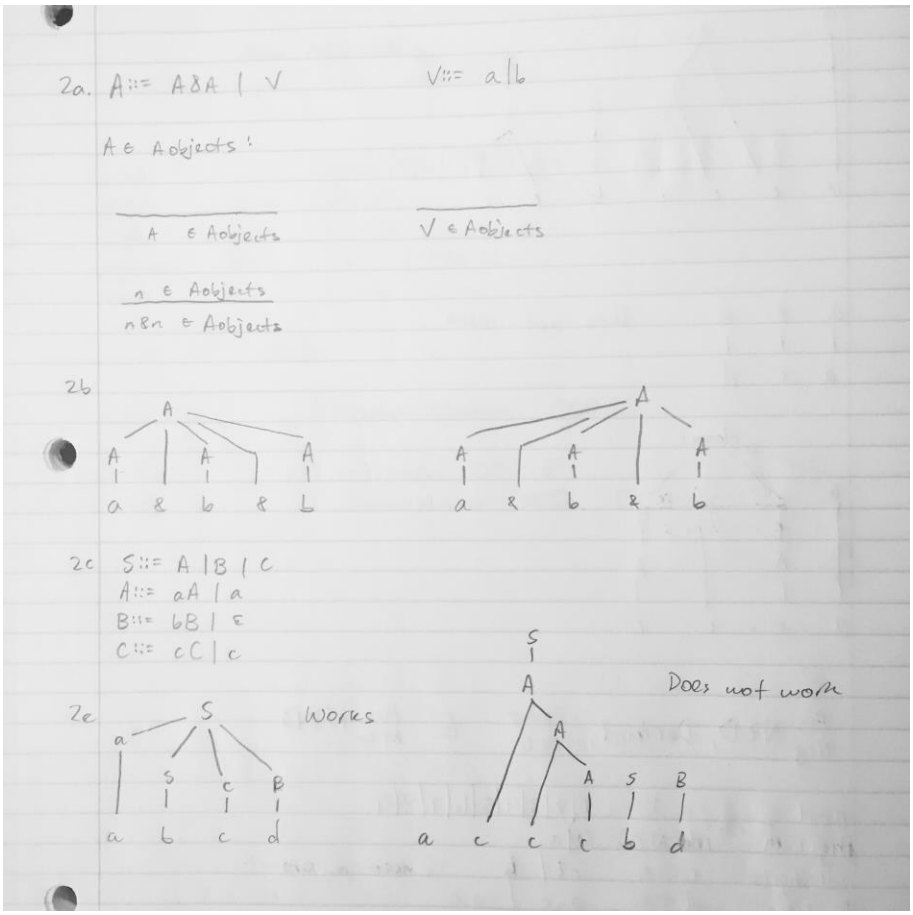
CSCI 3155

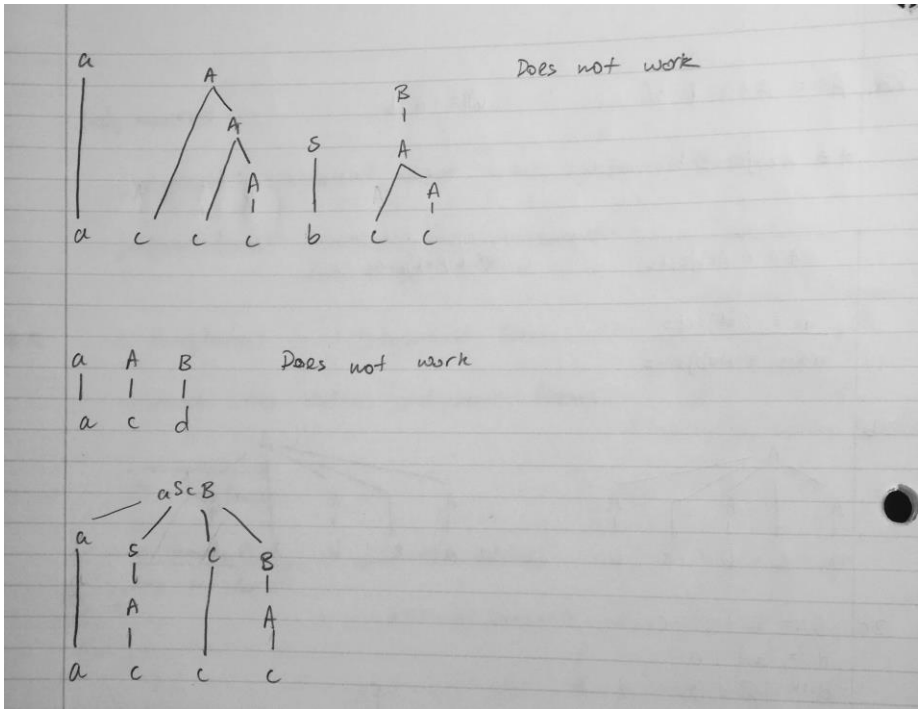
1.

2.

    a.

b.


2a. A ::= A&A | V
V ::= a|b
A ∈ Aobjects
A ∈ Aobjects
V ∈ Aobjects
n ∈ Aobjects
n&n ∈ Aobjects
2b
2c S ::= A | B | C
A ::= aA | a
B ::= bB | ε
C ::= cC | c
2e
Works
Does not work


Does not work
Does not work
a A B
a c d
aScB

c. L(S) is the set of strings that have any number of a's, any number of b's, any number of c's, or nothing.

L(S) = { "", "a^n", "b^n", "c^n": n in the set of Natural Numbers}

d.

i. S => AaBb (baa)

S => baBb

S => baab

ii. S => AaBb (bbbab)

S => AbaBb

S => AbbaBb

S => bbbaB => cannot go further

iii. S => AaBb(bbaaaaa)

S => AbaBb

S => bbaBb

S => bbaaB => cannot go further

iv. S => AaBb (bbaab)

S => AbaBb

S => bbaBb

S => bbaab

3.

a.

i. The first grammar generates an operand, or recursively, an operator operand. This means the expression will always begin with an operand, and can have an infinite number of operator operand sequences, ending with an operand.

   The second grammar starts with an operand followed by up to an infinite number of operator operand sequences, or an epsilon (nothing).

ii. These grammars generate the same expression because they both begin with an operand, then a variable number of operator operand sequences, and end with an operand.

b. Using 10 – 2 << 1 as an example:

10 – 2 << 1	=>		=>	16
(10 – 2) << 1	=>	8 << 1	=>	16
10 – (2 << 1)	=>	10 – 4	=>	6

Without parenthesis to force precedence, the scala interpreter evaluated the expression with "-" having a higher precedence than "<<".

c.

n	::=	1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
N	::=	N | nN
s	::=	ε | -N
E	::=	Esn | EsNn
z	::=	0 | 0z
fp	::=	z.z | sN.Nz | sz.NzE| sN.NzE