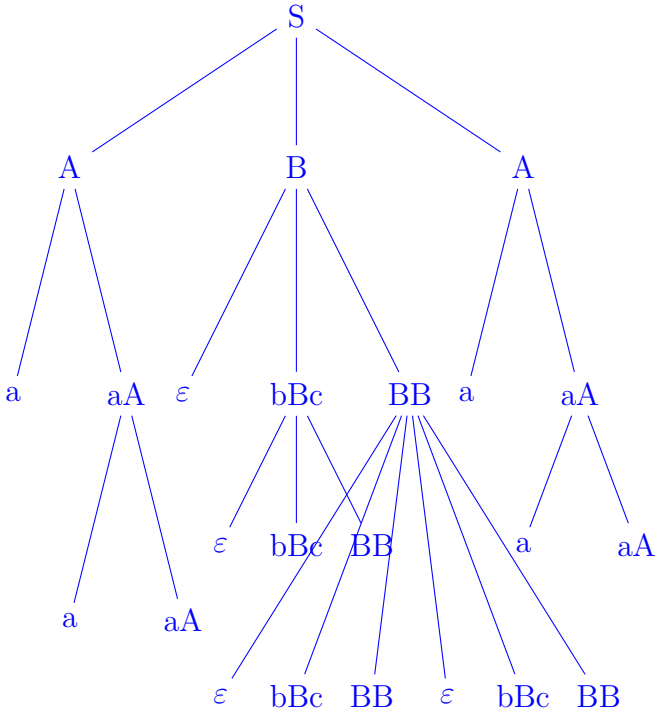
1. **Feedback**. Complete the survey linked from the moodle after completing this assignment. Any non-empty answer will receive full credit.

2. **Grammars: Synthetic Examples.**

   (a) Describe the language defined by the following grammar:

$$S ::= ABA$$
$$A ::= a|aA$$
$$B ::= \varepsilon|bBc|BB$$

This grammar defines a language that can produce a set of strings as small as 3 digits/values, all the way up to something with infinite values. The 'S' is called first and can produce a single digit/value of 'a' or 'aA' (which then another 'a' or 'aA' can be added to) for the first part. Then looks at 'B' and can return either a ε or 0, or 'bBc' (two strings and another call to 'B'), or 'BB' which is two more calls to 'B' that results in another case of ε/0, 'bBc', or 'BB'. Lastly, there is another call to 'A' which has the same outcomes as before and this can continue on for all calls to 'A' and 'B' until they are ended with single values of 'a' and/or ε/0.



(b) Consider the following grammar:

$$S ::= AaBb$$
$$A ::= Ab|b$$
$$B ::= aB|a$$

Which of the following sentences are in the language generated by this grammar? For the sentences that are described by this grammar, demonstrate that they are by giving **derivations**.

1. baab
2. bbbab
3. bbaaaaa
4. bbaab

(from Sebesta, Chapter 3)

Sentences 1 and 4 are in the language generated by this grammar.

Sentence 1 :

$$\alpha \Rightarrow \beta,$$
$$S \Rightarrow AaBb$$
$$S \Rightarrow baBb$$
$$S \Rightarrow baab$$

Sentence 4 :

$$\alpha \Rightarrow \beta,$$
$$S \Rightarrow AaBb$$
$$S \Rightarrow AbaBb$$
$$S \Rightarrow bbaBb$$
$$S \Rightarrow bbaab$$

(c) Consider the following grammar:

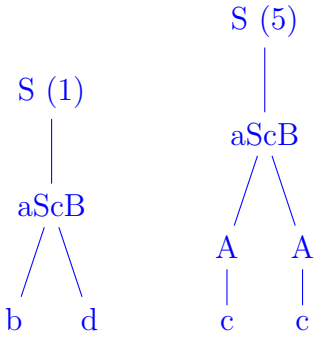
$$S ::= aScB|A|b$$
$$A ::= cA|c$$
$$B ::= d|A$$

which of the following sentences are in the language generated by this grammar? For the sentences that are described by this grammar, demonstrate that they are by giving **parse trees**.

1. abcd
2. acccbd
3. acccbcc
4. acd
5. accc

(from Sebesta, Chapter 3)

Sentences 1 and 5 are in the language generated by this grammar.



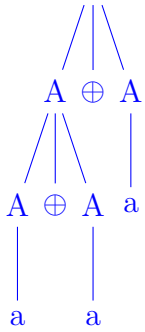
(d) Consider the following grammar:

$$A ::= a|b|A \oplus A$$

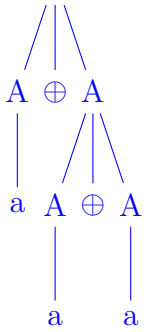
Show that this grammar is ambiguous.

Ambiguous grammar means that it can be "read" two different ways and so there can be multiple parse trees to get the same result. This grammar is ambiguous because if a=1, "1 + 1 + 1" could be written as both "1 + (1+1)" or "(1+1) +1".

A (left associative)



A (right associative)



(e) Let us ascribe a semantics to the syntactic objects A specified in the above grammar from part d. In particular, let us write

$$A \Downarrow n$$

for the judgment form that should mean A has a total n a symbols where n is the meta-variable for natural numbers. Define this judgment form via a set of inference rules. You may rely upon arithmetic operators over natural numbers. Hint: There should be one inference rule for each production of the non-terminal A (called a syntax-directed judgment form).

$$\frac{A_1 \Downarrow n_1; A_2 \Downarrow n_2}{A_1 \oplus A_2 \Downarrow n_1 + n_2}$$

3. **Grammars: Understanding a Language.**

   (a) Consider the following two grammars for expressions *e*. in both grammars, *operator* and *operand* are the same; you do not need to know their productions for this question.

$$e ::= operand | e\ operator\ operand$$

$$e ::= operand\ esuffix$$
$$esuffix ::= operator\ operand\ esuffix | \varepsilon$$

   i. Intuitively describe the expressions generated by the two grammars

   Both grammars start and end with "operand" and have alternating "operator" in the middle. Also, the first step 'operand' in the first grammar will be the last character in the first sentence, while the first step 'operand' in the second grammar will be the first character in the second sentence.

   ii. Do these grammars generate the same or different expressions? Explain.

   Both grammars generate the same expressions but are different in associativity. The first grammar is left associative, while the second grammar is right associative. The second grammar always has an appended ϵ at the end, but is removed in the final notation.

(b) Write a Scala expression to determine if '-' has higher precedence than '<<' or vice versa. Make sure that you are checking for precedence in your expression and not for left or right associativity. Use parentheses to indicate the possible abstract syntax trees, and then show the evaluation of the possible expressions. Finally, explain how you arrived at the relative precedence of '-' and '<<' based on the output that you saw in the Scala interpreter.

Scala expressions:

$$2 << 2 = 8$$
$$2 - 2 = 0$$

We would like to know if '<<' has higher precedence than '-', so we know by testing the Scala expressions:

$$2 - 2 << 2 = 0$$
$$(2 - 2) << 2 = 0$$
$$2 - (2 << 2) = -6$$

This shows us that the '-' operator has higher precedence than the '<<' operator, as Scala recognizes that we subtract the twos and then shift left by two.

(c) Give BNF grammar for floating point numbers that are made up of a fraction (e.g., 5.6 or 3.123 or -2.5) followed by an optional exponent (e.g.,E10 or E-10). The exponent, if it exists, is the letter 'E' followed by an integer. For example, the following are floating point numbers: 3.5E3, 3.123E30, -2.5E2, -2.5E-2, and 3.5. The following are not examples of floating point numbers: 3.E3, E3, and 3.0E4.5. More precisely, our floating point numbers must have a decimal point, do not have leading zeros, can have any number of trailing zeros, non-zero exponents (if it exists), must have non-zero fraction to have an exponent, and cannot have a '-' in front of a zero number. The exponent cannot have leading zeros. For this exercise, let us assume that the tokens are characters in the following alphabet Σ:

$$\Sigma \stackrel{\text{def}}{=} \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, E, -, .\}$$

Your grammar should be completely defined (i.e., it should not count on a non-terminal that it does not itself define).

$$S ::= I.N|I.NEI$$
$$I ::= B| - B$$
$$N ::= d|dN$$
$$B ::= c|cd$$
$$c ::= 1|2|3|4|5|6|7|8|9$$
$$d ::= 0|1|2|3|4|5|6|7|8|9$$