Jacob Berman

CSCI 3155 Lab 2 Write-up

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

S ::= *ABA*

A ::= a | a*A*

B ::= ε | b*B*c | *BB*

Answer:

- S is the start and has non-terminal symbols A and B.
- A will either produce an ‘a’ or an aA
- B will produce the epsilon which is empty, bBc, or BB

(b) Consider the following grammar:

S ::= *A*a*B*b

A ::= *A*b | b

B ::= a*B* | 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

TRUE

S ::= AaBb => baBb=> baab

2. bbbab

FALSE

S::= AaBb => AbaBb => AbbaBb => bbbaBb

- This fails because the non-terminal symbol B must produce at least one more ‘a’.

3. bbaaaaa

FASLE
- This fails because b is included in the start as a terminal above and therefore the statement must end with a b.

4. bbaab

TRUE

S::=AaBb => AbaBb => bbaBb => bbaab

(c) Consider the following grammar:

S ::= a*S*c*B* | *A* | b

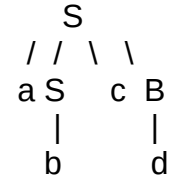
A ::= c*A* | c

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****.*

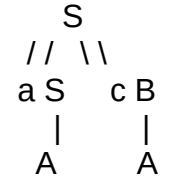
1. abcd

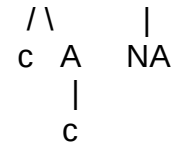
TRUE



2. acccbd

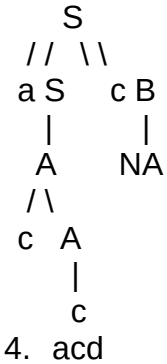
FALSE (cannot turn 'B' into 'bd'.)





3. acccbcc

FALSE because you can cannot turn 'B' into 'bcc'

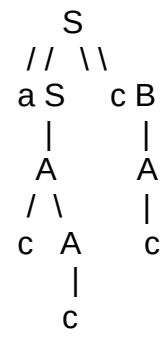


4. acd

FALSE
- Because the output is length 3. There is a value that returns empty set so this is false.

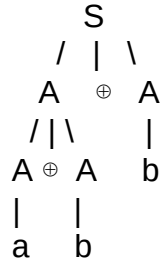
5. accc:

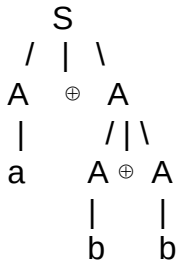
TRUE



(d) Consider the following grammar:

A ::= a | b | *A*⊕*A*





Show that this grammar is ambiguous.

(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 metavariable 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).

$$\frac{A \oplus A \Rightarrow a \;\&\; b,\ A \Rightarrow b}{(a \oplus b)b} \qquad \frac{A \Rightarrow b,\ A \Rightarrow a}{(b \oplus b)a}$$

3. ***Grammars:UnderstandingaLanguage****.*

(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 operand

e ::= operand esuffix

esuffix ::= operator operand esuffix | ε

i. Intuitively describe the expressions generated by the two grammars.

Answer: They both must start with an operand and end with an operand. They can have similar grammars.

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

Answer:

e ::= operand | e operator operand

e => operand

1. e => e operator operand operand => operand operator operand operand

e ::= operand esuffix

esuffix ::= operator operand esuffix | ε

1. e => operand esuffix => ε => operand
2. e => operand esuffix => operand operator operand esuffix =>
   operand operator operand

They can produce the same expression however they expand with different associativity. The first one expands to the left while the second expands to the right. The difference is if it adds the operand to the left or right.

(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.

Answer:

We discover the "-" has higher precedence over "<<" based off the scala interpreter. When using the expression 4-1 << 3, resulted in 24. When we add parentheses 4 – (1 << 3) and (4-1) << 3 and compare. The first expression gets, -4 and while the second gets 24 so this is how we know "-" has higher precedence.

(c) Give a 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 expo- nent, 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: def Σ = {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).

Answer:

S ::= YT.NE

Y::= - | ε

T ::= IN | ZN | ε

E ::= EYIN | ε

I ::=1,2,3…,9

Z ::= 0