Lab 2 Writeup

Michael Gohde

February 9, 2018

1 Overview

This document contains my responses to the questions written in Lab1.pdf.

2 Responses to questions

1. *Do the survey* Done
2. *Grammars: Synthetic Examples.*
   (a) *Describe the language defined by the following grammar:* (The aforementioned grammar)
   This grammar, at its topmost level, describes a tree consisting of three elements:
      i. an A, which may, in turn, form any combination of terminating 'a' or a terminating a in conjunction with another nonterminating 'A',
      ii. a nonterminating 'B', which may either take the form of nothing, a terminating 'b' followed by a nonterminating 'B' followed by a terminating 'c' or two consecutive nonterminating 'B's,
      iii. another nonterminating 'A' as described above.
      In general, statements generated by this grammar start and end with a sequence of terminating 'a's with either a set of 'b's and 'c's in the middle or nothing.

iv. *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***

A. 'baab' Should be possible. See the end of this problem for a derivation.

B. 'bbbab' is not possible, since there must be at least two As in the middle; B evaluates to either aB or a, and it is a required part of the grammar.

C. 'bbaaaaa' is also not possible, since the base grammar is terminated with a 'b'.

D. 'bbaab' should be possible, since A may evaluate to Ab, which can then evaluate to bb. There are enough as and the string terminates with a b.

```
Derivation for baab:

S -> AaBb -> baBb -> baab

Derivation for bbaab:

S -> AaBb -> AbaBb -> bbaBb -> bbaab
```

v. *Consider the following grammar. Which of the following sentences are in the language based on this grammar? For the sentences that are described by this grammar, demonstrate that they are by giving parse trees:*

A. 'abcd' is possible:

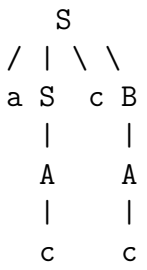
```
      S
    / |\ \
    a S c B
      |   |
      b   d
```

B. 'acccbd' is not possible. b cannot come after c, since only S may generate a 'b'

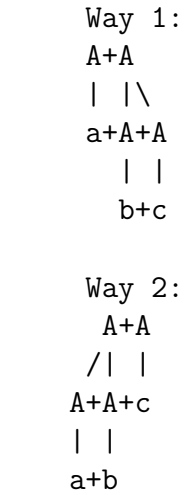
C. 'acccbcc' is not possible for the reason stated above.

D. ‘acd’ is not possible, since S cannot take any form that wouldn’t generate a character between ‘a’ and ‘c’

E. ‘accc’ should be possible:



vi. *Show that this grammar is ambiguous* Given the sentence ‘a+b+c’, the tree may be constructed in more than one way:



Since more than one syntax tree can be ascribed to the same sentence, there is ambiguity.

vii. *Let us ascribe semantics to the syntatic object A specified in d.*

(b) *Grammars: Understanding a language*

i. *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.*

A. *Intuitively describe the expressions enerated by the two*

grammars. The first grammar should generate a chain of operands interspersed with operators. The first and last elements are always operands. It is possible for there to be a single operand.
The second grammar should also generate chains of operands interspersed with operators. In fact, both grammars should be equivalent, since they both describe languages consisting of operands interspersed with operators.

ii. *Write a scala expression to determine if '-' has higher precedence than 'ji'.*
One such expression may be:

```
1<<4-2
```

If the left shift operator has higher precedence, then this expression should evaluate to 14. Else, it should evaluate to 4.
The scala interpreter gives the following result:

```
scala> 1<<4-2
res0: Int = 4
```

As such, Scala places higher priority on the subtraction operator than on the left shift operator. This means that the expression can be rewritten as the following with no change in effects:

```
1<<(4-2)
```

iii. *Give a BNF grammar for floats* For this exercise, I will use 'sig' for big sigma. Please note that E is not a non-terminating experession. Rather, it is th literal 'E' defined in the alphabet.

```
F ::= 0.RT | eps.RT | -0.RT | -eps.RT
R ::= epsR | eps
T ::= EepsR | E-epsR
```

(c) *Implement the lab.* Done!