Jake Traut

# Lab 2 Write-Up

2. a)  $A ::= A \& A \mid V$
       $V ::= a \mid b$  $\Longrightarrow$   $A \in \text{Aobjects}$
       $V \in \text{Vobjects}$
       $a, b \in \text{Vobjects} \rightarrow a, b \in \text{Aobjects}$

b) Show the grammar is ambiguous. Will use two different parse trees to get to same result "aab".
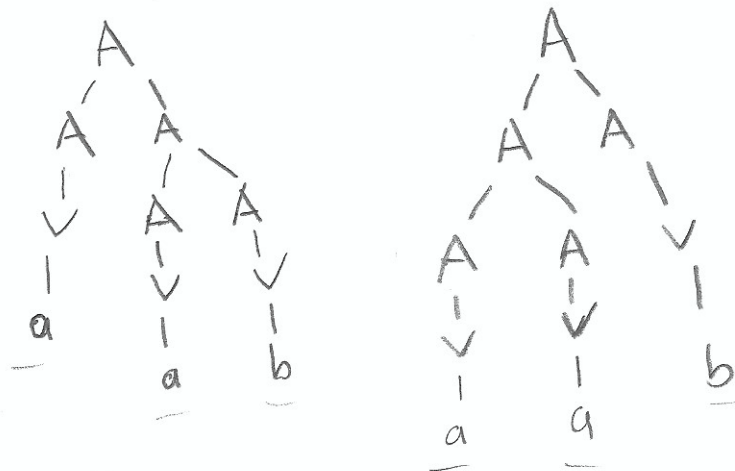


c) Describe the language defined by the following grammar:

$S ::= A \mid B \mid C$
$A ::= aA \mid a$
$B ::= bB \mid \varepsilon$
$C ::= cC \mid c$

Will create a string of repeating letters, S defines what letter the string will be, a, b, or c, no mixing. A, B, or C can then be called repeatedly to make a string of any size, the only exception or difference being B has the possibility of making an empty string "" due to the terminating character $\varepsilon$ that defines empty.

2. d)  $S ::= AaBb$
$A ::= Ab | b$
$B ::= aB | a$

1. baab   is valid
$S \Rightarrow AaBb$
$\Rightarrow baBb$
$\Rightarrow baab$ ✓

2. bbbab not valid, can't have only 1 a.
3. bbaaaaa not valid, must have b at end.
4. bbaab is valid
$S \Rightarrow AaBb$
$\Rightarrow AbaBb$
$\Rightarrow bbaBb$
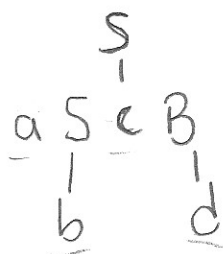$\Rightarrow bbaab$

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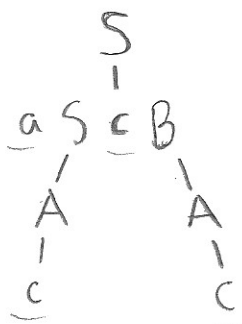$A ::= cA | c$
$B ::= d | A$

1. abcd is valid



2. acccbd not valid, no b after c
3. acccbcc not valid, b placement
4. acd not valid, aScB will force another character between a and c
5. accc is valid

**3. a)**

1) e ::= operand | e operator operand

2) e ::= operand esuffix
   esuffix ::= operator operand esuffix | ε

1)
```
        e
        |
   e operator operand
   |
e operator operand
|
operand
```

2)
```
        e
        |
   operand esuffix
            |
    operator operand esuffix
                     |
           operator operand esuffix
                              |
                              ε
```

Both grammars can be called repeatedly to make an expression of any number of operator calls on the operand, and they generate the same expressions, just terminated differently, the second uses ε as empty so doesn't change the expression.

i.e: operand operator operand operator operand

**b) Precedence check**

$$\text{if} ((1<<2-1) == ((1<<2)-1)) \text{ return } "<<"$$
$$\text{else if} ((1<<2-1) == (1<<(2-1))) \text{ return } "-"$$

In the scala worksheet this expression evaluates to "–" so the minus has precedence

3. c) Floating Point grammar.

$$F ::= SI_1 . I_r \mid SI_1 . I_r E SI_1 \mid 0$$

$$S ::= - \mid \varepsilon \; (\text{Empty/blank for "+"})$$

$$I_1 ::= D_1 I_r \mid D_1$$

$$D_1 ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

$$I_r ::= D_r I_r \mid D_r$$

$$D_r ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$