Homework 3a Solutions
CSc 4330/6330 Spring 2018

1. Consider the alphabet {a, b, c} and the language over this alphabet consisting of strings with the following properties:

   - a's appear before b's and b's appear before c's, and
   - the number of a's, b's, and c's are equal.

   Examples of strings in the language are aaabbbccc, aabbcc, abc, etc. Write an attribute grammar for this language. You will solve this problem in 2 steps:

   - Step 1: Write a context free grammar for the language in which the a's appear before b's and the b's appear before c's. This grammar need not consider the second condition that the number of a's, b's, and c's are equal (In fact it is impossible to construct a context free grammar to satisfy the second condition).

     S →ABC
     A →aA | ε
     B →bB | ε
     C →cC | ε

   - Step 2: Introduce attributes for non-terminals in your grammar and define the attribute computation functions and predicates to enforce the second condition.

   Attributes: length, synthesized, associated with non-terminals A, B, C, and stores the length of the string generated from the non-terminal

   Syntax rule: S → ABC
   Predicate: (A.length == B.length) and (B.length == C.length)

   Syntax Rule: A → ε
   Semantic Rule: A.length = 0

   Syntax Rule: A → aA
   Semantic Rule: A[1].length ← 1+A[2].length

   Syntax Rule: B → ε
   Semantic Rule: B.length = 0

   Syntax Rule: B → bB
   Semantic Rule: B[1].length ← 1+B[2].length

   Syntax Rule: C → ε
   Semantic Rule: C.length = 0

   Syntax Rule: C → cC
   Semantic Rule: C[1].length ← 1+C[2].length

2. Consider the following grammar:

```
btree : NIL | LPAREN btree COMMA value COMMA btree RPAREN
value : digit | digit value
digit : 0|1|2|3|4|5|6|7|8|9
```

Some examples of binary trees are:

```
(NIL,12,NIL)
((NIL,12,NIL),20,(NIL,15,NIL))
((NIL,3,NIL),8,((NIL,12,NIL),20,(NIL,15,NIL)))
```

Augment this grammar with attributes and the attribute computation functions and predicates to accept binary trees that are "balanced". A balanced binary tree is one in which the heights of the subtrees at each interior node are within one of each other.

Attributes height: synthesized; associated with btree; stores height of tree generated by non-terminal

Syntax rule: btree → NIL
Semantic Rule: btree.height ← 0

Syntax rule: btree -> LPAREN btree COMMA value COMMA btree RPAREN
Semantic Rule: btree[1].height ← 1+max(btree[2].height,btree[3].height)
Predicate: abs(btree[2].height-btree[3].height)<=1

Syntax rule: value : digit | digit value

Syntax rule: digit : 0|1|2|3|4|5|6|7|8|9

3. Write denotational semantics for Boolean expressions defined in the following grammar:

```
bool : TRUE | FALSE | var | boolExpr
boolExpr : leftExpr bop rightExpr | NOT leftExpr
leftExpr : TRUE | FALSE | var
rightExpr : TRUE | FALSE | var
bop : AND | OR
```

You can assume that a VARMAP function is available that will contain boolean values for boolean variables. If variables have value **undef** the denotational semantics should return **error**.

Universe of objects: {true, false}

```
Mb(bool,s) =
  case bool of
    TRUE => true
    FALSE => false
    var => if (VARMAP(var,s)==undef)
             then error
             else VARMAP(var,s)
    boolExpr =>
      case boolExpr of
        leftExpr bop rightExpr =>
          if (Mb(leftExpr,s)==error ||
              Mb(rightExpr,s)==error)
          then error
          else if (bop==AND) then
              Mb(leftExpr,s)&&Mb(rightExpr,s)
          else
              Mb(leftExpr,s)||Mb(rightExpr,s)
        NOT leftExpr =>
          if (Mb(leftExpr,s)==error)
          then error
          else !Mb(leftExpr,s)

Mb(leftExpr,s) =
  case leftExpr of
    TRUE => true
    FALSE => false
    var => if (VARMAP(var,s)==undef)
             then error
             else VARMAP(var,s)
```

Similar definition for Mb(rightExpr,s)

Note: Here &&, ||, and ! are boolean operators on boolean values true and false.