Homework 1

September 11, 2018

For this homework, you will write BNF grammars. You do not need to resolve ambiguity such as operator precedence or associativity. Your grammars do not consider white spaces.

Assume that there exists the non-terminal `<num>` that represents numbers.

1. Define a non-terminal `<var>` that represents the variables a, b, c, x, y, and z.

2. Define a non-terminal `<aop>` that represents binary arithmetic operators `+, -, *, /, ^`.

3. Define a non-terminal `<aexp>` that represents arithmetic expressions. The grammar should accept expressions such as: `((a - 5) / c + 1) * 2`, `2 * (x + 3) * y + z / a`, and `2^44 + b`. Please utilize the non-terminals such as `<var>`, `<num>`, and `<aop>`.

4. Define a non-terminal `<bop>` that represents binary boolean operators `&&` and ||. Note that '||' is quoted since | is also used as separator for the grammar.

5. Define a non-terminal `<cop>` that represents binary comparison operators `<,<=,==,!=,>=,>`

6. Define a non-terminal `<bexp>` that represent logical expressions (i.e. boolean expressions). The grammar should accept expressions such as:

   ```
   !(a/2 <= 10 && 5 != c)
   a < c || b > 0
   ```

   Note that logical expressions include negation operator ! and the boolean constant **true** and **false**. You should utilize the non-terminals `<aexp>`, `<bop>`, `<cop>`.

7. Define a non-terminal `<Stmt>` that represents statements that include assignments, while loops, and if-statements (with optional else-part.) The statements may include arithmetic, comparison, and boolean expressions.

   Define a non-terminal `<Stmts>` that represents zero or more statements.

Note that `<Stmt>` and `<Stmts>` are mutually dependent.

You may use the special non-terminal `<empty>` to represent nothing.

You should utilize the non-terminals `<var>`, `aexp`, `<bexp>`.

Here is program that can be represented by the non-terminal `<Stmts>`.

```
b = true;
y = 2;
while(!b && y < x) {
   if ((y - (y/x)*x) == 0) {
      b = false;
   }
}
if(b) {
  z = y/x;
}
else {
  z = 1;
}
```

Here is another program that can be represented by `<Stmts>`.

```
x = 1;
y = x;

if (a > x) {
  while (x <= a) {
    x = x + 1;
    y = y * x;
  }
}
```