

# Homework 1: Defining BNF Grammars

Stephen Wagstaff

CS 431

September 13, 2018

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

```
<var> ::= 'a' | 'b' | 'c' | 'x' | 'y' | 'z'
```

2. Define a non-terminal `<aop>` that represents binary arithmetic operators.

+, -, \*, /, ^.

```
<aop> ::= '+' | '-' | '*' | '/' | '^'
```

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>`.

```
<aexp> ::= '(' <aexp> ')'
        | <aexp> <aop> <aexp>
        | <num> | <var>
```

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.

```
<bop> ::= '&&' | '||'
```

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

```
<cop> ::= '<' | '>' | '<=' | '>=' | '==' | '!='
```

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>`.

```
<bexp> ::= '!' <bexp> ')'
        | <bexp> <bop> <bexp>
        | <aexp> <cop> <aexp>
        | 'true' | 'false'
```

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>`.

```
<Stmt> ::= <var> '=' (<bexp> | <aexp>) ';'
        | 'while(' <bexp> ')' {' <Stmts> '}'
        | 'if(' <bexp> ')' {' <Stmts> '}' ['else {' <Stmts> '}']
```

```
<Stmts> ::= <Stmts> <Stmt> | <empty>
```

The BNF `<Stmt>` definition can be found at the end of the document

---

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;
    }
}

```

---

```

<Stmt> ::= <var> '=' <aexp> ';'
        | <var> '=' <bexp> ';'
        | 'while(' <bexp> ')' {' <Stmts> '}'
        | 'if(' <bexp> ')' {' <Stmts> '}'
        | 'if(' <bexp> ')' {' <Stmts> '}' 'else {' <Stmts> '}'

```