**CS 381 Homework 3 – Syntax**

Submit a pdf for problems 1 – 4 and a Haskell *.hs file for problem 5.

1. Using the grammar below, show a parse tree and a leftmost derivation for the sentence

A = B * (C+A)

.

<assign> → <id> = <expr>

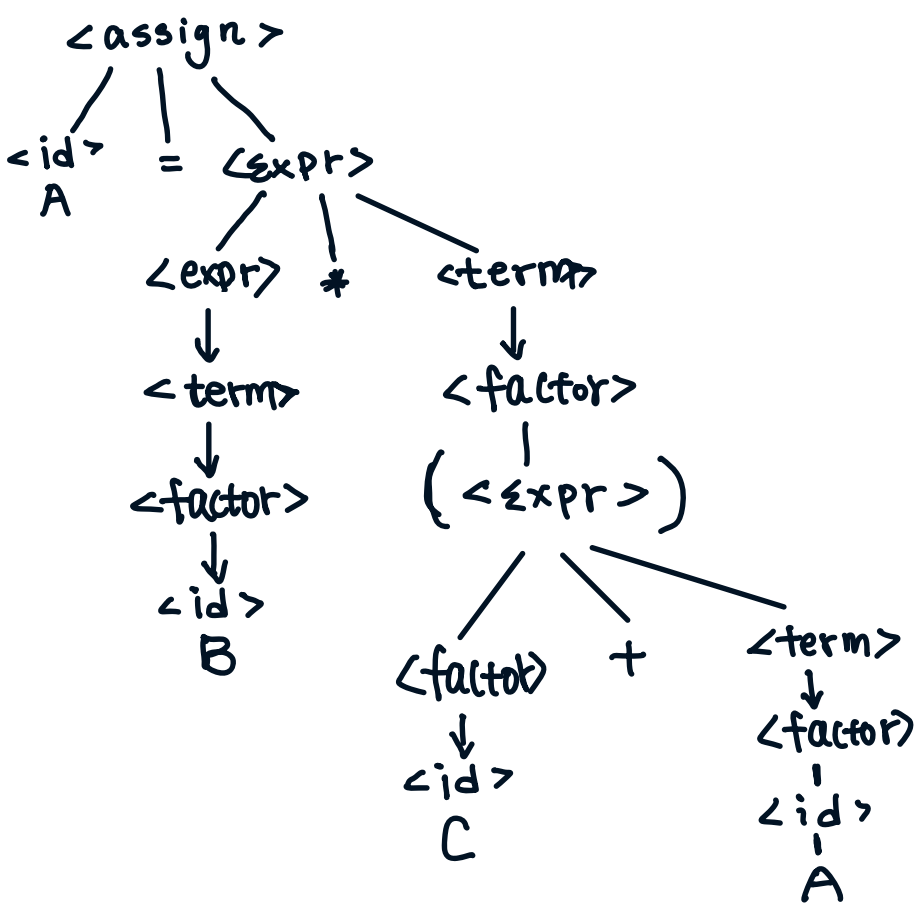
<expr> → <expr> * <term>
| <term>

<term> → <factor> + <term> | <factor> - <term>
| <factor>

<factor> → ( <expr> )
| <id>

<id> → A | B | C

Parse tree:



## Lefmost derivation:

assign $\Rightarrow$ <id> = <expr>

$\Rightarrow$ A = <expr>

$\Rightarrow$ A = <expr> * <term>

$\Rightarrow$ A = <term> * <factor>

$\Rightarrow$ A = <factor> * (<expr>)

$\Rightarrow$ A = <id> * (<factor> + <term>)

$\Rightarrow$ A = B * (<id> + <factor>)

$\Rightarrow$ A = B * ( C + <id> )

$\Rightarrow$ A = B * ( C + A )

2. Rewrite the following BNF to add the prefix ++ and -- unary operators of Java.

<assign> → <id> = <expr>

<expr> → <expr> * <term>

| <term>

<term> → <factor> + <term> | <factor> - <term>

| <factor>

<factor> → ( <expr> )

| <id>

<id> → A | B | C

added the prefix and -- unary operators ;

<assign> → <id> = <expr>

<expr> → <expr> * <term>

| <term>

<term> → <factor> + <term> | <factor> - <term>

| <factor>

<factor> → ( <expr> )

| <id> | ++<id> | --<id> | ++<factor>

| -- <factor>

<id> → A | B | C

3. Show that the following grammar is ambiguous

.

<compare> → <boolexpr> == <boolexpr>

<boolexpr> → <boolexpr> **AND** <boolexpr>

| <boolexpr> **OR** <boolexpr>

| <bool>

| **NOT** <bool>

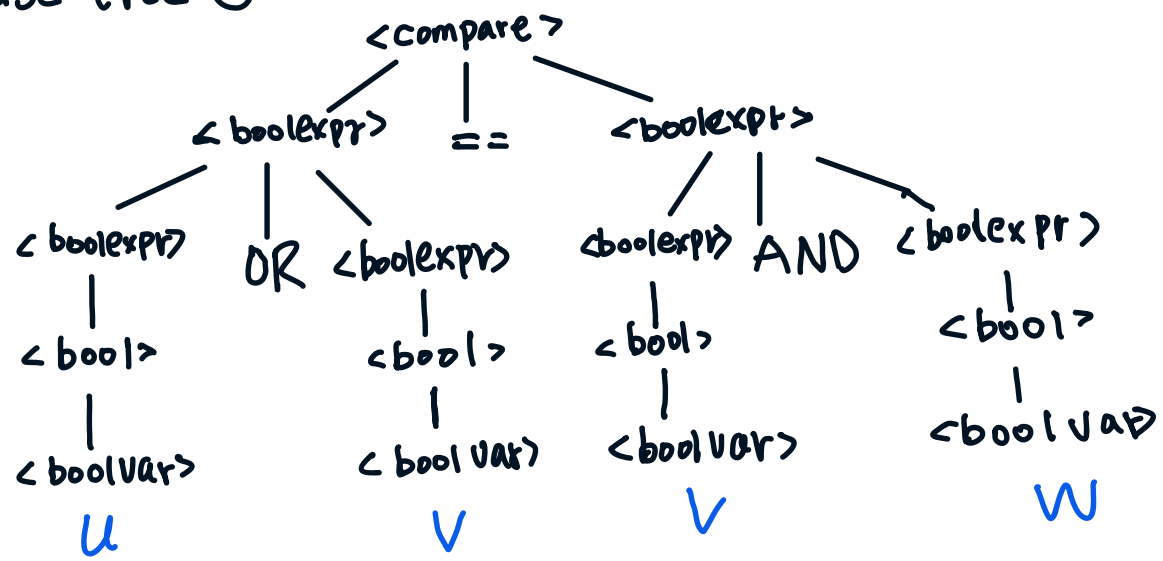
<bool> → <boolvalue> | <boolvar>

<boolvalue> → **True** | **False** | **0** | **1**

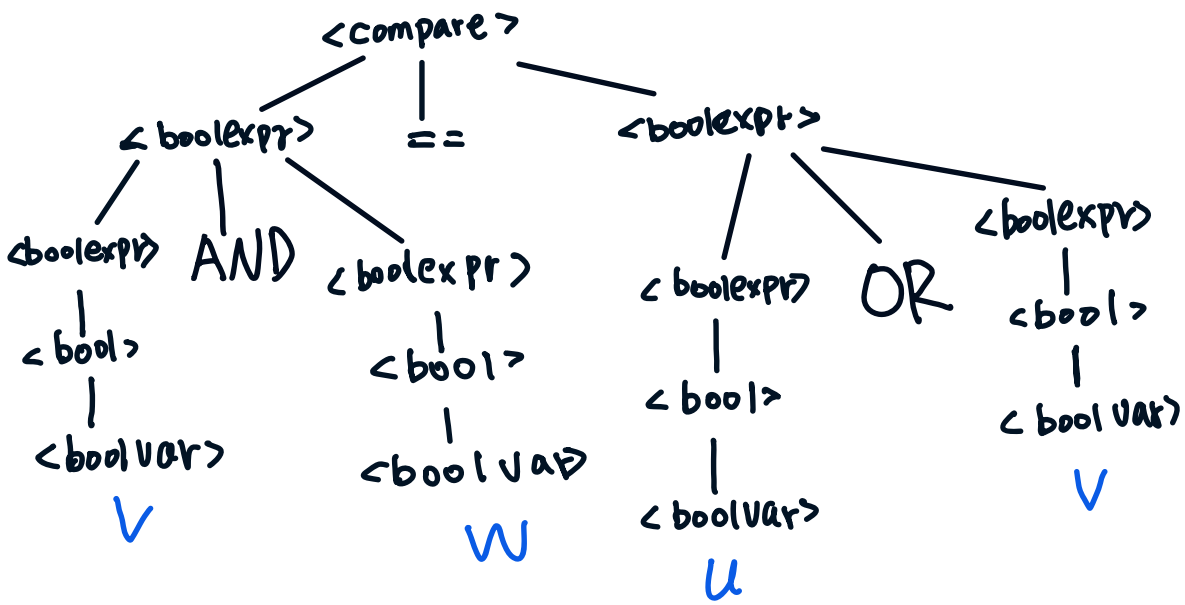
<boolvar> → **u** | **v** | **w**

Let me generate two parse trees for the sentence : u OR V == V AND W

Parse tree ① :



Parse tree ②:




The first parse tree represents the sentence U OR V == V AND W and the second parse tree represents sentence V AND W == U OR V. Therefor, The grammer is ambiguous, Because the same iput sentence could be parsed in two different ways.

4. Write a grammar G for the language L consisiting of strings of 0's and 1's that are the binary representation of odd integers greater that 4. For example 11 $\notin$L, 101$\in$L, 110 $\notin$L. Draw parse trees for the strings 1011 and 1101

Let $G = (\{V\}, \{0,1\}, P, S)$, where S is the start symbol, $V=\{S,A\}$ is set variable, $\{0,1\}$ is set of terminals, and P is the set of production rules defined as :

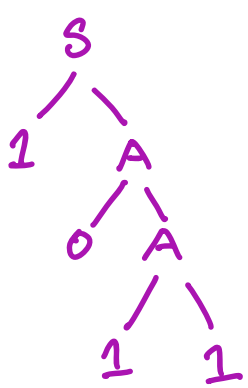
$S \rightarrow 1A$

$A \rightarrow 0A$

$A \rightarrow 1$

This grammar genearates binary representation of odd integers greater than 4, where the first character is always 1, followed by zero or more 1's and 0's alternating, ending with a 0.

Parse tree for the string "1011"




Parse tree for the string "1101"