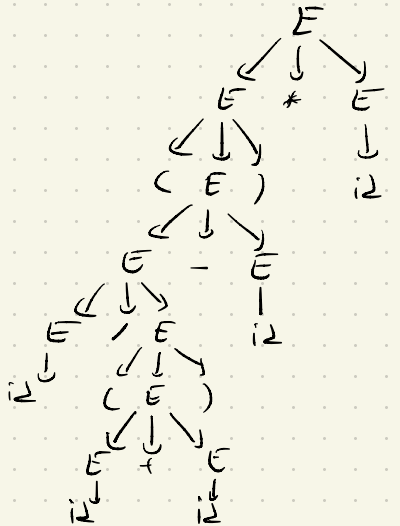
HOMEWORK 4

1a) Context Free Gramar (CFG):-

E -> (E) | E+E | E-E | E*E | E/E | id

This is grammar for calculator. One rule for +, −, *, /, an ( ). For every operator its needed two operands ∀ The use of operator and operands must be E operator E. For parentheses have in between two parentheses have expression so rule is (E).

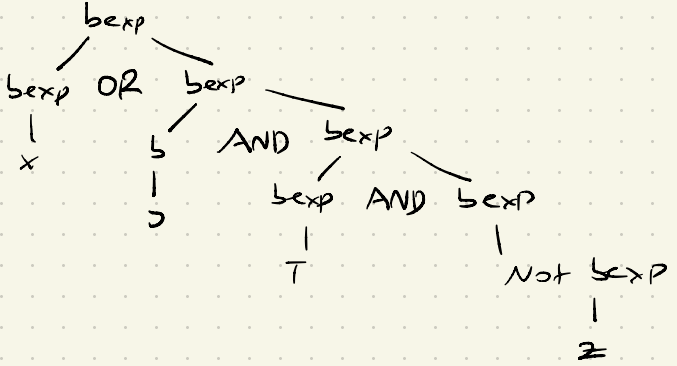
b) Parse tree for (0.23 / (5+3.1) − 20) * 2



-> This gives (id / (id+id) - id) * id .

= (0.23 / (5+3.1) - 20) * 2

2-) $E = x \vee y \wedge (T \wedge \neg z)$



CFG for it:

bexp → bexp OR bexp | bexp AND bexp

bexp → NOT bexp

bexp → TRUE | x | y | z

bexp → FALSE | x | y | z

Exercise 5.1.2

b) 1001

Leftmost	Rightmost
$S \to A1B$	$S \to A1B$
$\to 11B$	$\to A10B$
$\to 101B$	$\to A100B$
$\to 100B$	$\to A1001B$
$\to 1001B$	$\to A1001$
$\to 1001$	$\to 1001$

c) 00011

Leftmost	Rightmost
$S \to A1B$	$S \to A1B$
$\to 0A1B$	$\to A11B$
$\to 00A1B$	$\to A11$
$\to 000A1B$	$\to 0A11$
$\to 0001B$	$\to 00A11$
$\to 00011B$	$\to 000A11$
$\to 00011$	$\to 00011$

Exercise 5.1.3

Base case: Starting with single characters, with no operations. To consider a single character a, we can just have a CFG with a single transition: $S \to a$.

Inductive step: Suppose that for any regex with fewer than k operators we can build a CFG that generates the same language. Then take a language L be represented by the regex R, such that R has k operators.

Three possible operations:

$+, \cdot, *$

For union, if $R = R_1 + R_2$ then assume that R_1 is generated by CFG with start symbol S_1, and that R_2 is generated by a CFG with start symbol S_2. We create a new start state S with the production rule $S \to S_1 | S_2$. If the first production we use is $S \to S_1$, then we will produce exactly the strings matched by R_1, by our IH. If the first production we use is $S \to S_2$, then we will produce matchings with R_2.

For concatenation, if $R = R_1 R_2$, then we use the same basic as union.

For Kleene star, if $R = R_1^*$, then we create a new CFG with start state $S \to S_1 S \mid \epsilon$. Then we can prove by induction that a string with any number of copies of strings matched by R_1 can be accepted.