Problem Set 1: BNF

Abstract

This problem set's main theme is centered around BNF. A short and brief description of BNF starts the problem set. The rest of the problem set is creating BNF grammars according to some given specifications. From those grammars we draw parse trees of given sentences from the task language.

1) **Task 1 - BNF?**

Backus-Naur form, or Backus normal form, is a formal syntax used mostly for describing context-free grammars and programming languages. BNFs are useful in that they can explicitly describe the syntax or grammar of a language. A BNF grammar consists of tokens, nonterminal symbols, productions/rules, and a nonterminal start symbol. With a BNF we can draw parse trees and examine how sentences(expressions of a language) are produced using our production/rewriting rules. A parse tree should start with

the start symbol, use the productions to produce a given sentence of tokens.

2) **Task 2 - BNF Description of L1**

Start symbol = L1

Non-terminals = {L1, no-string, plus-string, minus-string, plus, minus}

Tokens = { (,), +, -}

Productions:

<L1> ::= <no-string> | <plus-string> | <minus-string>

<no-string> ::= () | () <L1>

<plus-string> ::= (<plus>) | (<plus>) <L1>

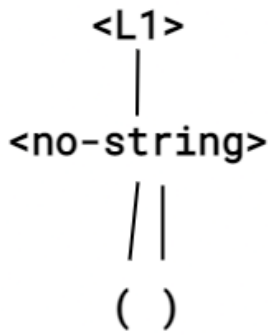
<minus-string> ::= (<minus>) | (<minus>) <L1>

<plus> ::= + | + <plus>

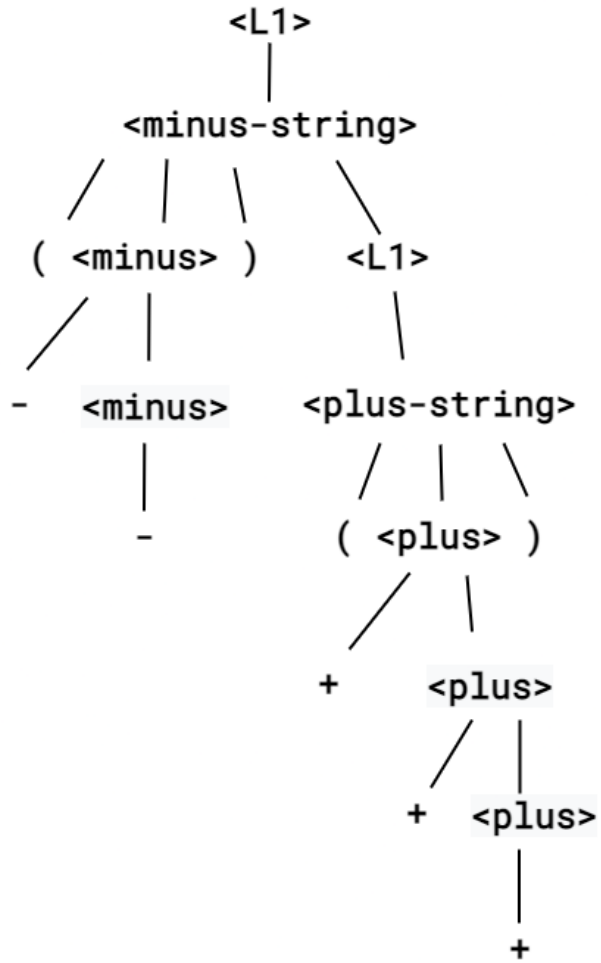
<minus> ::= - | - <minus>

3) **Task 3 - Parse Trees for L1**

1. ()



2. (--)(+++)



4) **Task 4 - BNF Description of L2**

Start symbol = num

Non-terminals = {num, zero, qn, qd}

Tokens = {0, 1, 2, 3}

Productions:

<num> ::= <zero>

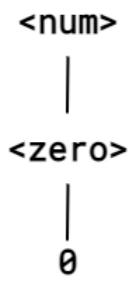
<zero> ::= 0 | <qn>

<qn> ::= <qd> | <qd> <qn>

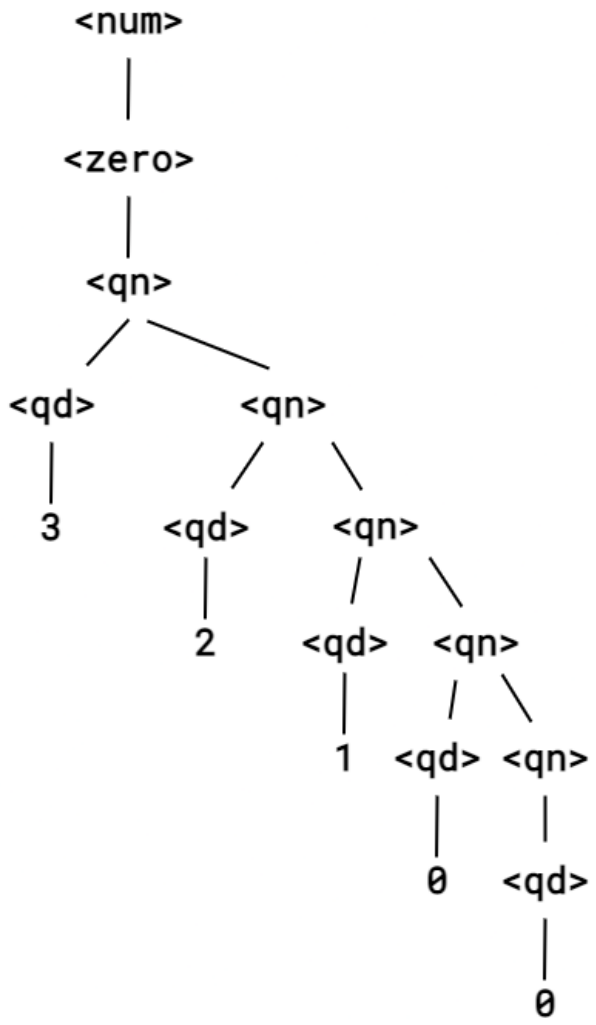
<qd> ::= 0 | 1 | 2 | 3

5) **Task 5 - Parse Trees for L2**

1. 0



2. 32100



6) **Task 6 - BNF Description of L3**

Start symbol = bool-exp

Non-terminals = {bool-exp, op-exp, exp-list, exp, const-list, const}

Tokens = { (,), and, or, not, #f, #t }

Productions:

<bool-exp> ::= <const> | <op-exp>

<op-exp> ::= (and <exp-list>) | (or <exp-list>) | (not <exp>)

<exp-list> ::= <op-exp> <exp-list> | <const-list> <exp-list> | empty

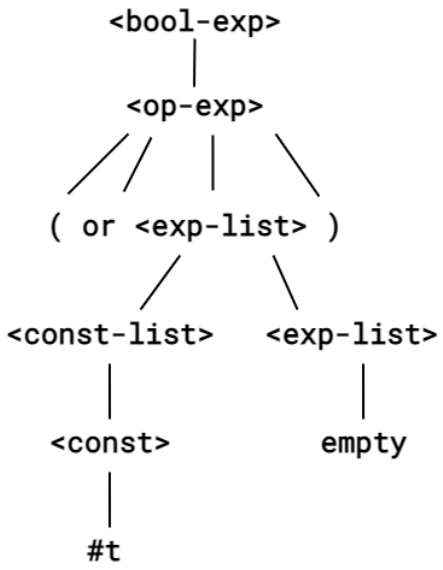
<exp> ::= <op-exp> | <const>

<const-list> ::= <const> | <const> <const-list>

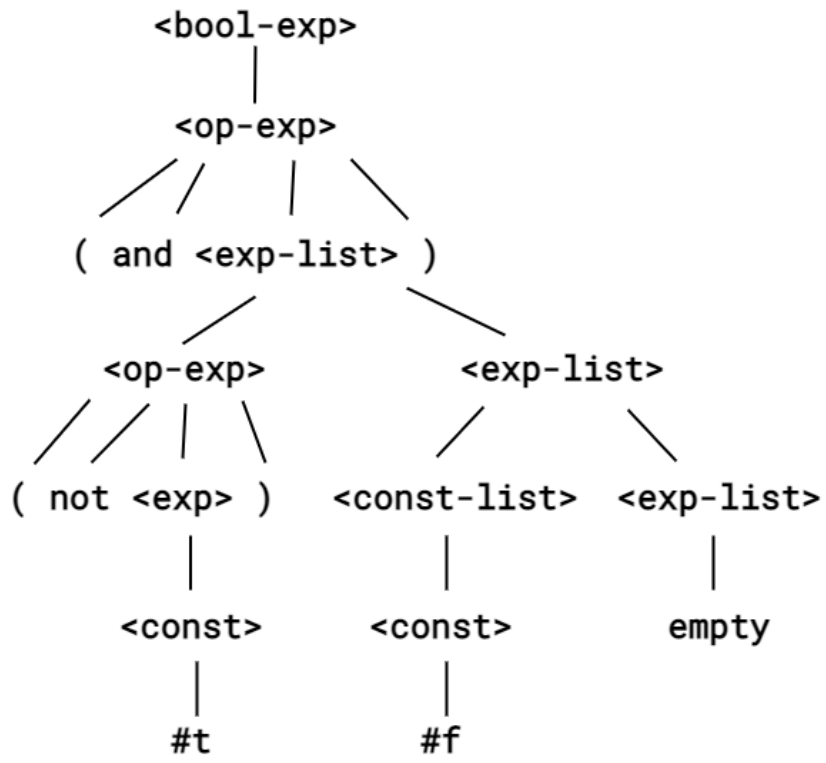
<const> ::= #f | #t

7) **Task 7 - Parse Trees for L3**

1. (or #t)



2. (and (not #t) #f)



8) **Task 8 - BNF Description of L4**

Start symbol = L4

Non-terminals = {L4, HL, TL, ones, tens, aft-ten}

Tokens = {zero, one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen, seventeen, eighteen, nineteen, twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety, hundred}

Production:

<L4> ::= <HL> | <TL> | <ones> | <aft-ten> | zero

<HL> ::= <ones> hundred | <ones> hundred <TL> | <ones> hundred <ones> | <ones> hundred <aft-ten>

<TL> ::= <tens> | <tens> <ones> | ten

<tens> ::= twenty| thirty | forty | fifty | sixty | seventy | eighty | ninety

<ones> = one | two | three | four | five | six | seven |

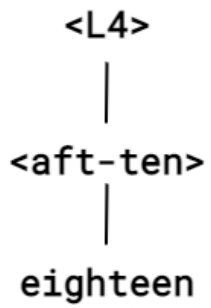
eight | nine

<aft-ten> = eleven | twelve | thirteen | fourteen | fifteen

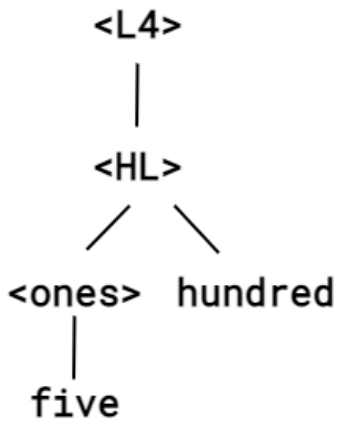
| sixteen | seventeen | eighteen | nineteen

9) **Task 9 - Parse Trees for L4**

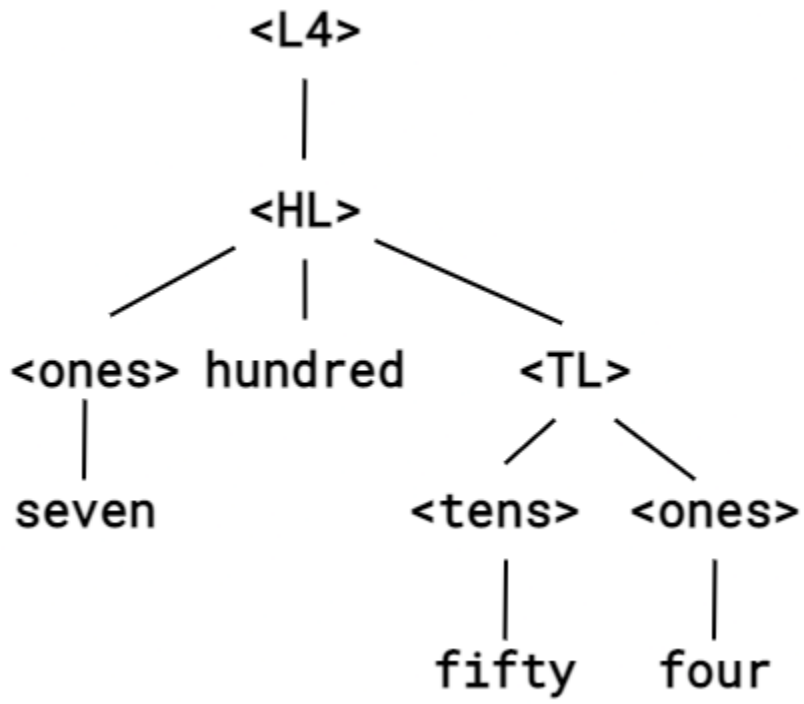
1. eighteen



2. five hundred



3. seven hundred fifty four



10) Task 10 - BNF Description of L5

Start symbol = cf

Non-terminals = {cf, add-exp, show-exp, describe-exp,

color-id, num}

Tokens = {0,1,2, ..., 255, colors, add, color, show,(,)}

Productions:

<cf> ::= <add-exp> | colors | <show-exp> | <describe-exp>

<add-exp> ::= add (<num> <num> <num>) <color-id> |

add (<num> <num> <num> <num>) <color-id> | add color

<color-id>

<show-exp> ::= show <color-id>

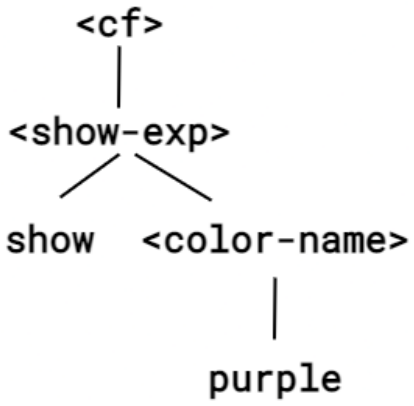
<describe-exp> ::= describe <color-id>

<num> ::= 0 | 1 | 2 | ... | 255

11) Task 11 - Parse Trees for L5

1. colors

2. show purple



3. add (100 220 170) c28