Temitope Emokpae 2/13/2023

CSC 344

First Problem Set Assignment: BNF

Learning Abstract

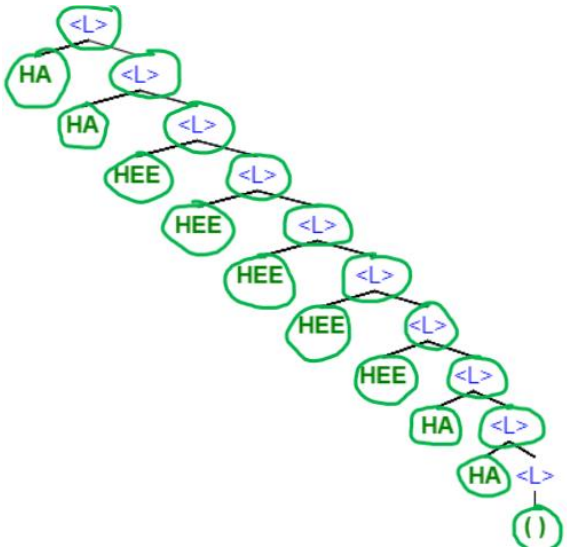
This assignment features defining BNF and its functions. It also demonstrates making a parse tree using the BNF created.

Problem 1: Laughter

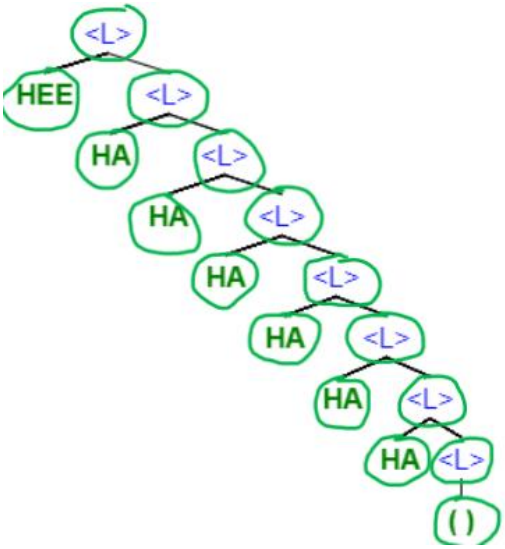
BNF Description

<L> ::= HA < L > | HEE <L> | <empty>

Parse Tree for Laughter (1)



Parse Tree for Laughter (2)

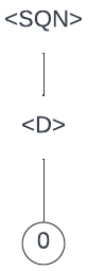


Problem 2: SQN (Special Quaternary Numbers)

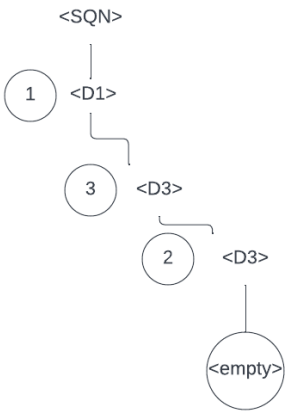
BNF Description

<SQN> ::= <empty> | <D> | 0 <D0> | 1 <D1> | 2 <D2> | 3 <D3>
<D> ::= 0 | 1 | 2 | 3
<D0> ::= 1 <D1> | 2 <D2 > | 3 <D3> | <empty>
<D1> ::= 0 <D0> | 2 <D2> | 3 <D3> | <empty>
<D2> ::= 0 <D0> | 1 <D1> | 3 <D3> | <empty>
<D3> ::= 0 <D0> | 1 <D1> | 2 <D2> | <empty>

Parse Tree for SQN (1)



Parse Tree for SQN (2)



Parse Tree for SQN (3)

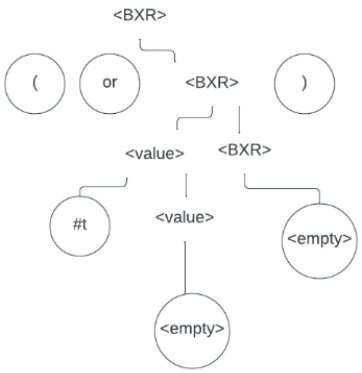
The string “1223” cannot be drawn because the production rule for <D2> won’t allow another 2 to come after.

Problem 3: BXR

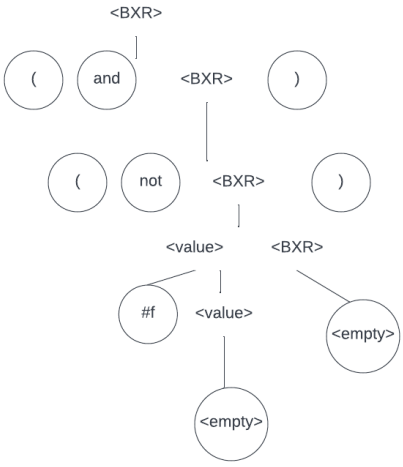
BNF Description

< BXR> ::= (and <BXR>) | (or <BXR>) | (not #f) <BXR> | (not #t) <BXR> | <value> <BXR> | <empty>
<value> ::= #t <value> | #f <value> | <empty>

Parse Tree for BXR (1)



Parse Tree for BXR (2)

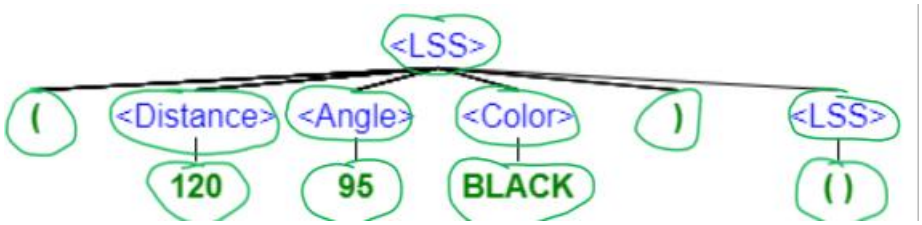


Problem 4: LSS (Line Segment Sequences)

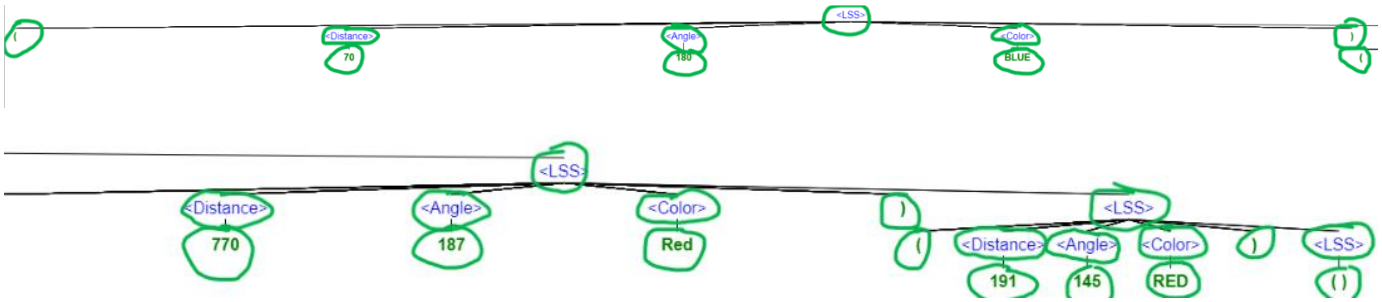
BNF Description

<LSS> ::= (<Distance> <Angle> <Color>) <LSS> | <empty>
<Color> ::= RED | BLACK | BLUE

Parse Tree for LSS (1)



Parse Tree for LSS (2)

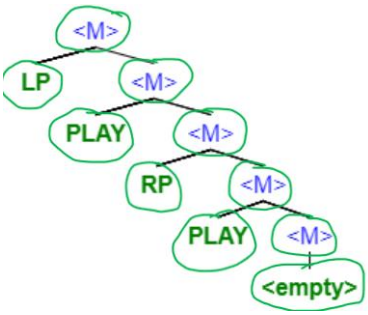


Problem 5: M-Lines

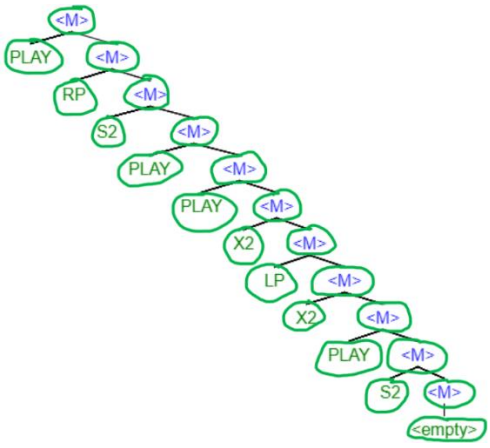
BNF Description

<M> ::= PLAY <M> | REST <M> | RP <M> | LP <M> | S2 <M> | X2 <M> | <empty>

Parse Tree for M-Lines (1)



Parse Tree for M-Lines (2)



Problem 6: BNF?

BNF Definition

BNF, short for Backus–Naur Form, was created by a program designer named John Bakus. It's the formal technique used in computer science for structuring and describing the grammar syntax of any programming language. It contains sets of terminal symbols, nonterminal symbols, tokens, and the production rule. The production rule gets used to represent the left-hand nonterminal side being replaced by what is on the right-hand terminal or nonterminal side. After that, a parse tree gets constructed that breaks down the long expression into tiny parts based on the BNF made for it.