Joseph Scollo

CSC 344

BNF Assignment

Abstract:

The focus of this assignment is to demonstrate the method in which one could structure and refine a BNF grammar to create a useful or meaningful programing language. BNF grammar is governed by a “start” symbol followed by sets of tokens, non-terminal symbols and productions(rewrite rules). Several small languages with differing syntax and constraints will have their grammar rules formally demonstrated and will be formally tested via the use of parse trees.

Problem 1: Laughter

Consider language Laughter which consists of strings of any number of the symbols HA and HEE subject only to the constraint that strings of the HA symbol must be even in length and sequences of the HEE symbol must be odd in length.

Examples:

1. HA HA
2. HEE
3. HA HA HEE HEE HEE HA HA HEE HA HA HA HA
4. HEE HA HA HEE HEE HEE HA HA HEE

Start: { Laughter }

Tokens: { HA, HE }

Non-Terminals: { Laughter, laugh1, laugh2, laugh3, Empty}

Production Rule Set:

<Laughter> ::= <laugh1> | <laugh2> | <Empty>

<laugh1> ::= HA HA | HA HA <laugh1> | HA HA <Laughter>

<laugh2> ::= HEE | HEE <laugh1> | HEE <laugh3>

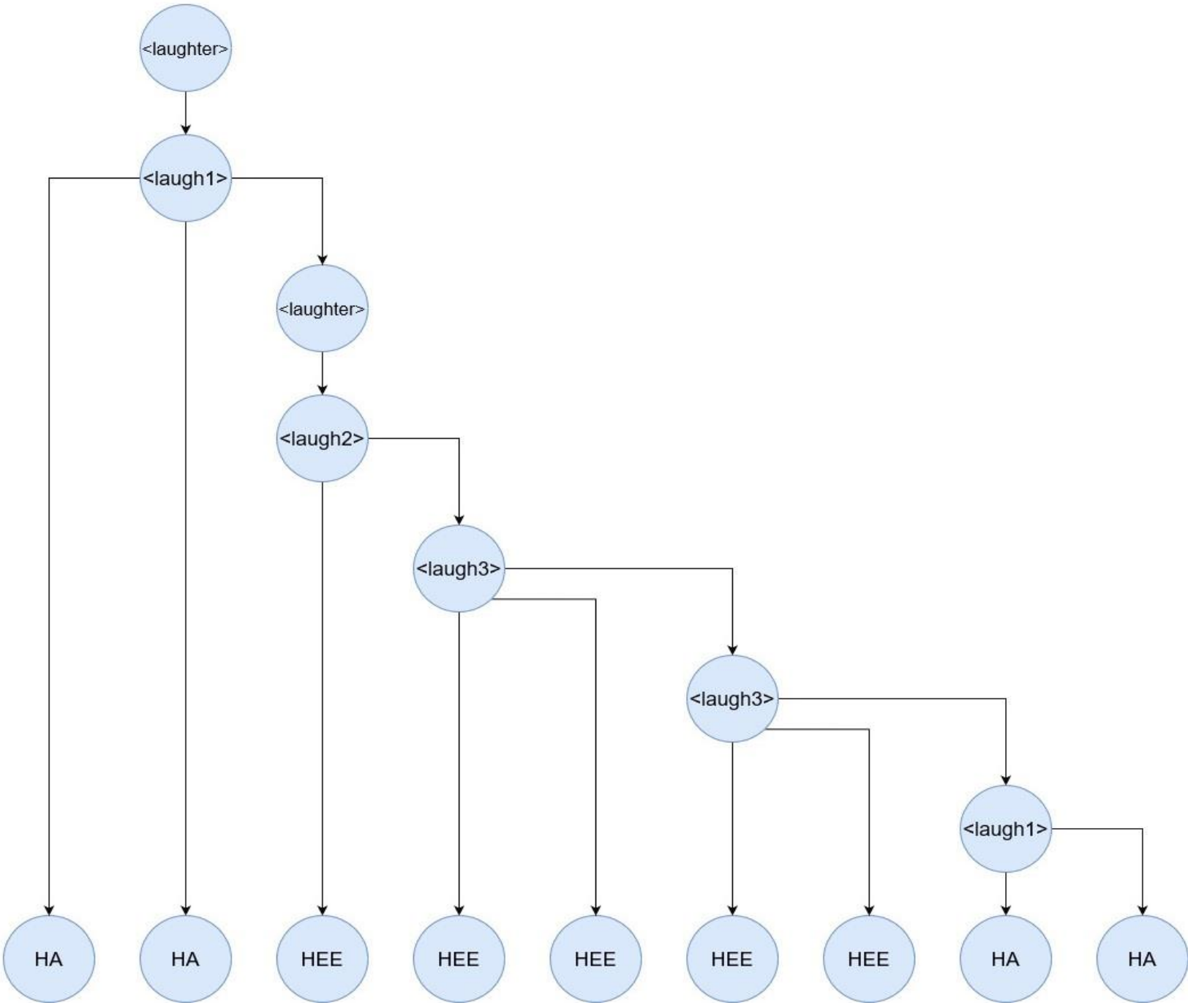
<laugh3> ::= HEE HEE | HEE HEE <laugh3> | HEE HEE <laugh1> | <Laughter>

The above rule set will be used to generate the parse trees for following sentences:

1. HA HA HEE HEE HEE HEE HEE HA HA
2. HEE HA HA HA HA HA HA

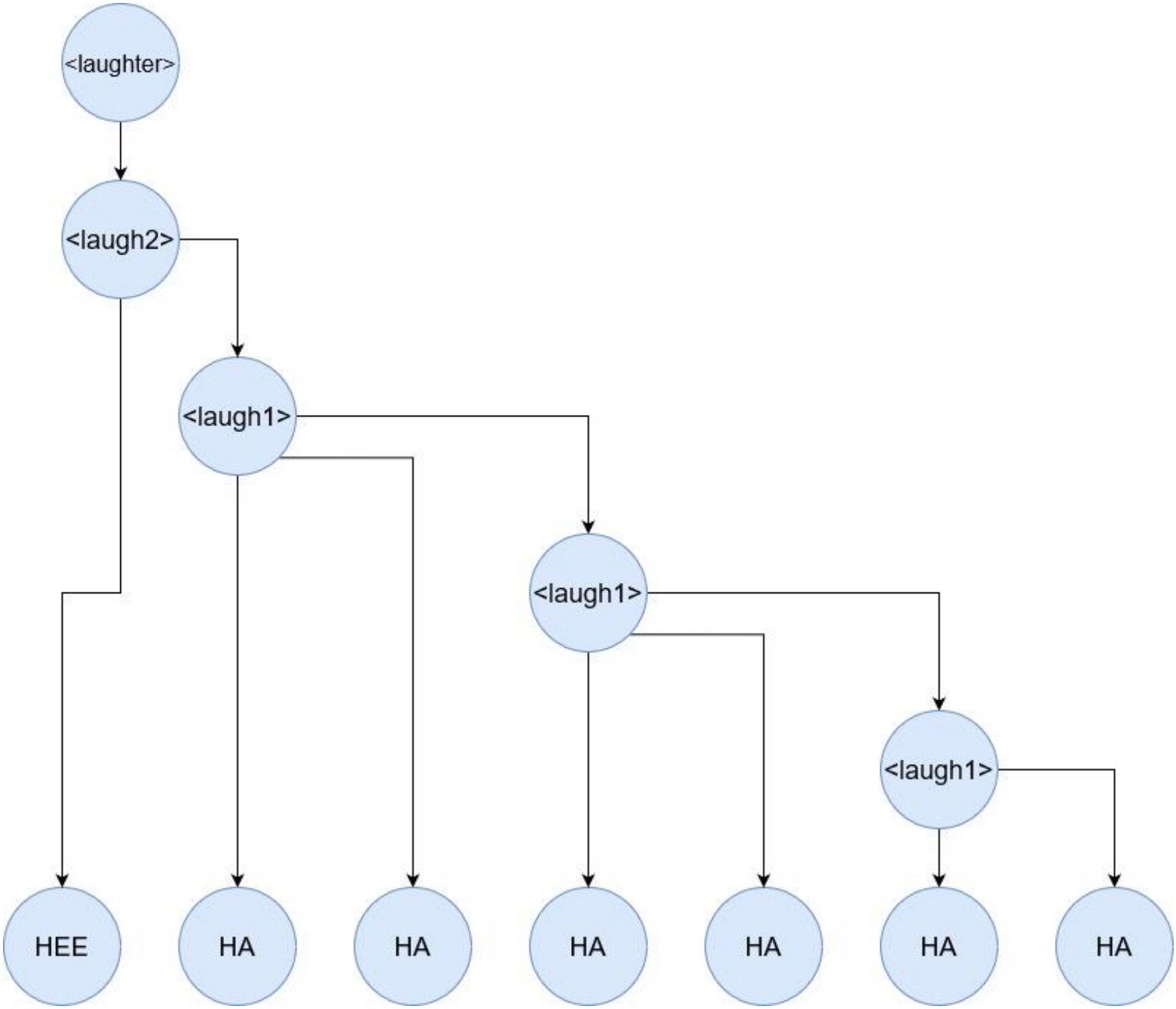
Parse Tree 1:

HA HA HEE HEE HEE HEE HEE HA HA



Parse Tree 2:

HEE HA HA HA HA HA HA



Problem 2: SQN (Special Quaternary Numbers)

Consider the language SQN which consists of the set of all quaternary numbers with no leading zeros, and with no two adjacent occurrences of the same quaternary digit:
For example, the following are sentences in this language:

1. 0
2. 123012301230
3. 10121320212303132

The following strings, for example, are not sentences in this language:

1. 1233
2. 010
3. 12322221

Start: { QN }

Tokens: { 0, 1, 2, 3 }

Non-Terminals: { QN, QD, NZQD, QDS, Empty }

Production Rule Set:

<QN> ::= 0 | <NZQD>

<NZQD> ::= 1 | 2 | 3 | 1 <QD_SET1> | 2 <QD_SET2> | 3 <QD_SET3> |

<QD_SET1> :: = 0 <QD_SET0> | 2 <QD_SET2> | 3 <QD_SET3> | <END>

<QD_SET2> :: = 0 <QD_SET10> | 1 <QD_SET1> | 3 <QD_SET3> | <END>

<QD_SET3> :: = 0 <QD_SET0> | 1 <QD_SET1> | 2 <QD_SET2> | <END>

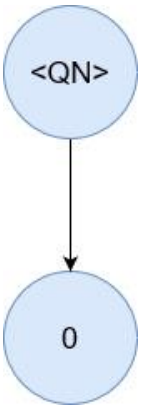
<QD_SET0> :: = <NZQD> | <END>

<END> :: = <Empty>

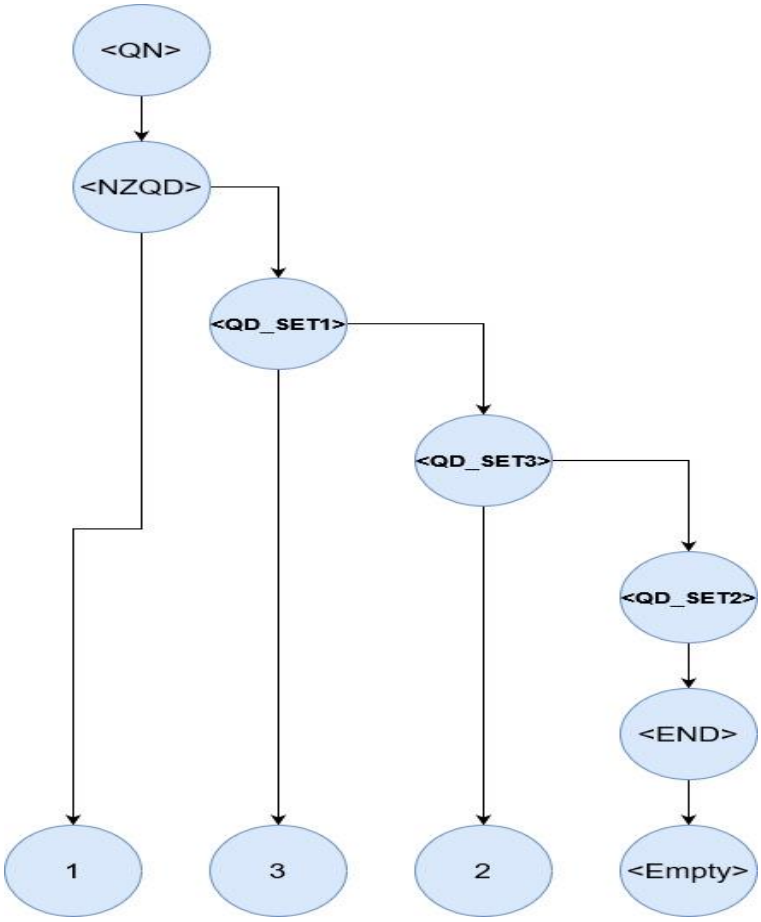
The above rule set will be used to generate the parse trees for following sentences:

1. 0
2. 132

Parse Tree 1:



Parse Tree 2:



The string 1223 in not legal or possible because the production rules of the grammar are not crafted in a way that would produce repeating adjacent quaternary digits.

Problem 3: BXR

Consider language BXR to be the set of Boolean valued expressions in Racket which are composed of the constants #t and #f and just the operators and, or and not.

Start: { BXR }

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

Non-Terminals: { ops, consts, empty }

Production Rule Set:

<BXR> ::= <ops> | <consts>

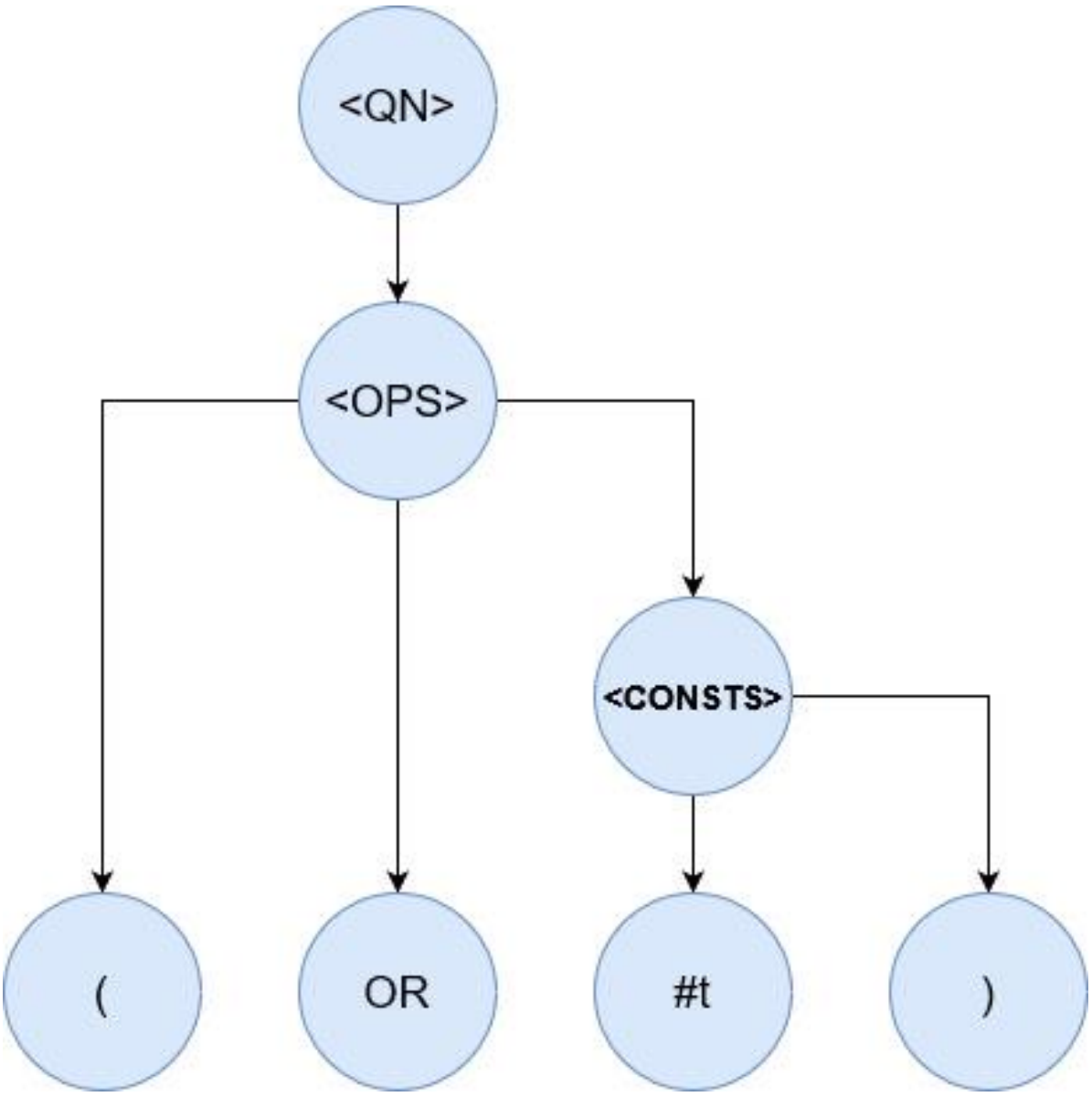
<ops> ::= (and <conts> | (or <consts> | (not #t) <ops> | (not #f) <ops> | (and
<ops> | or <ops> | (not #t) <const> | (not #f) <const>

<consts> :: = #t) | #t (<ops> | #f (<ops> | #f)

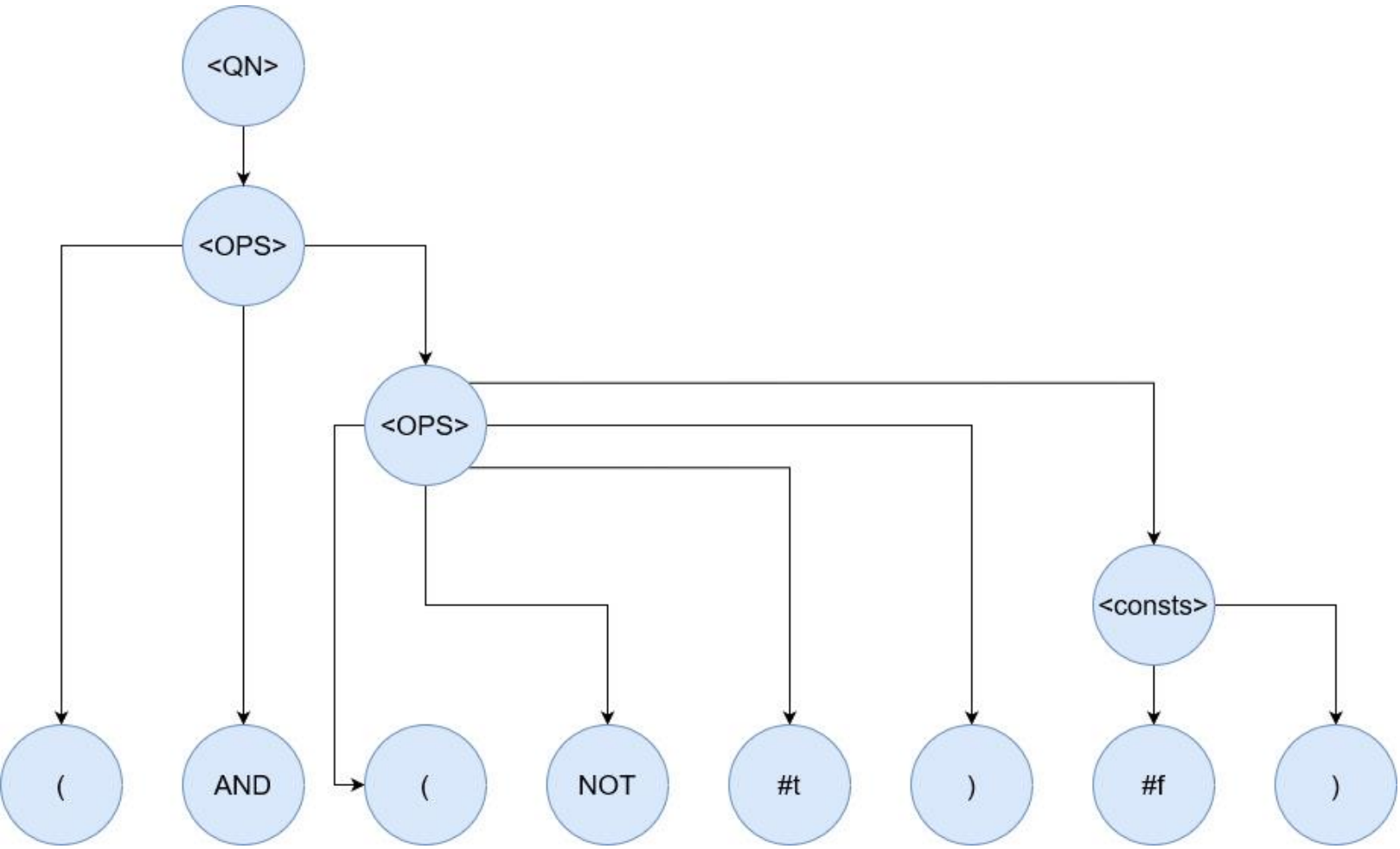
The above rule set will be used to generate the parse trees for following sentences:

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

Parse Tree 1:



Parse Tree 2:



Problem 4: LSS (Line Segment Sequences)

Consider the language of strings of zero or more parenthesized triples, each consisting of a positive integer representing a distance, a positive integer representing an angle, and the name of a color, either RED or BLACK or BLUE.

Examples:

1. (100 90 BLUE) (50 0 RED) (200 270 BLACK)
2. (5 270 BLACK) (10 280 BLACK) (15 290 BLACK) (20 300 BLACK)

Start: { LSS }

Tokens: { (,), RED, BLUE, BLACK, SEGMENT}

Non-Terminals: { COLORS, ANGLE , DISTANCE, Empty }

Production Rule Set:

<LSS> ::= <Empty> |

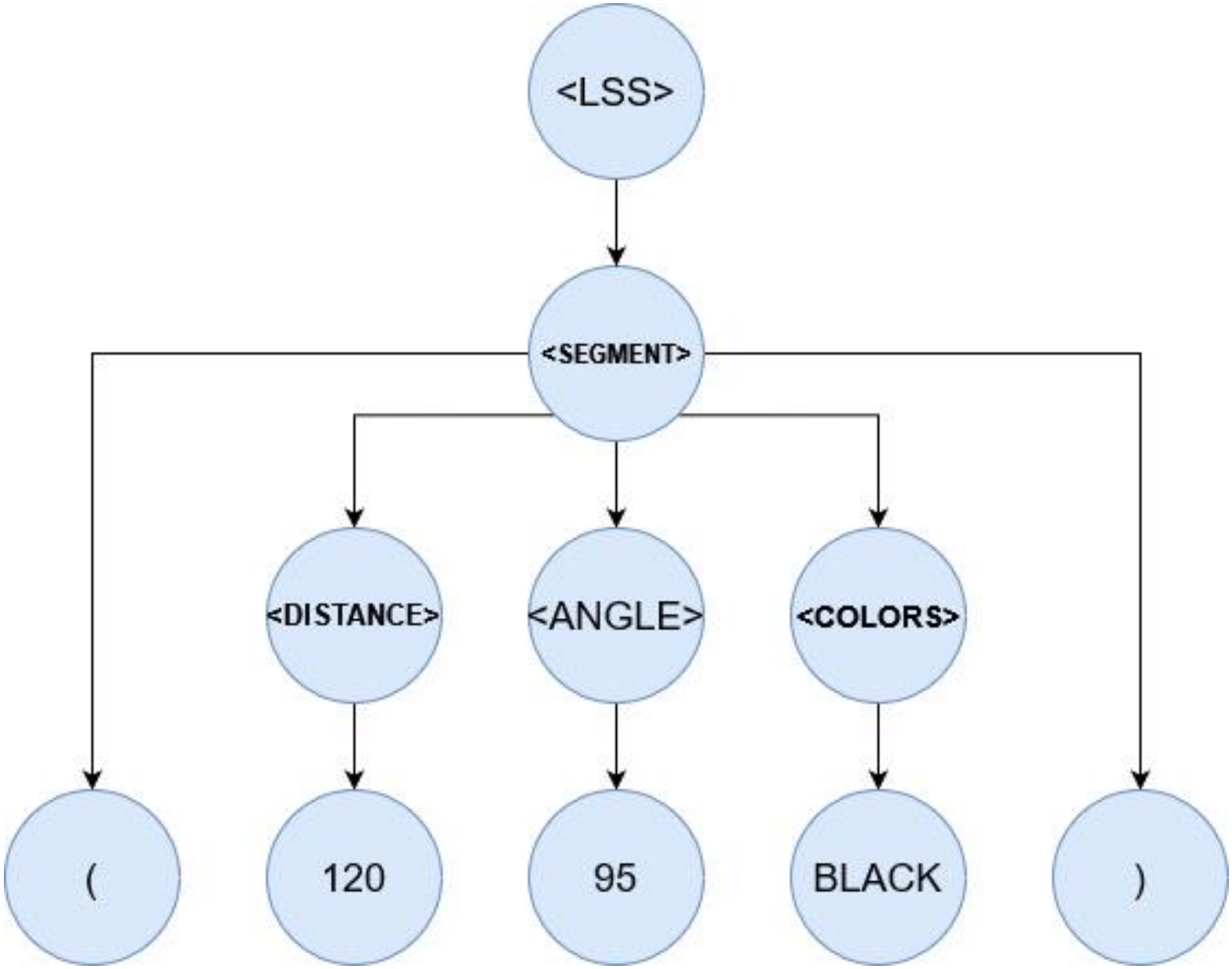
::= (<DISTANCE> <ANGLE> <COLORS>) | (<DISTANCE>
<ANGLE> <COLORS>)

<COLORS> ::= RED | BLUE | BLACK

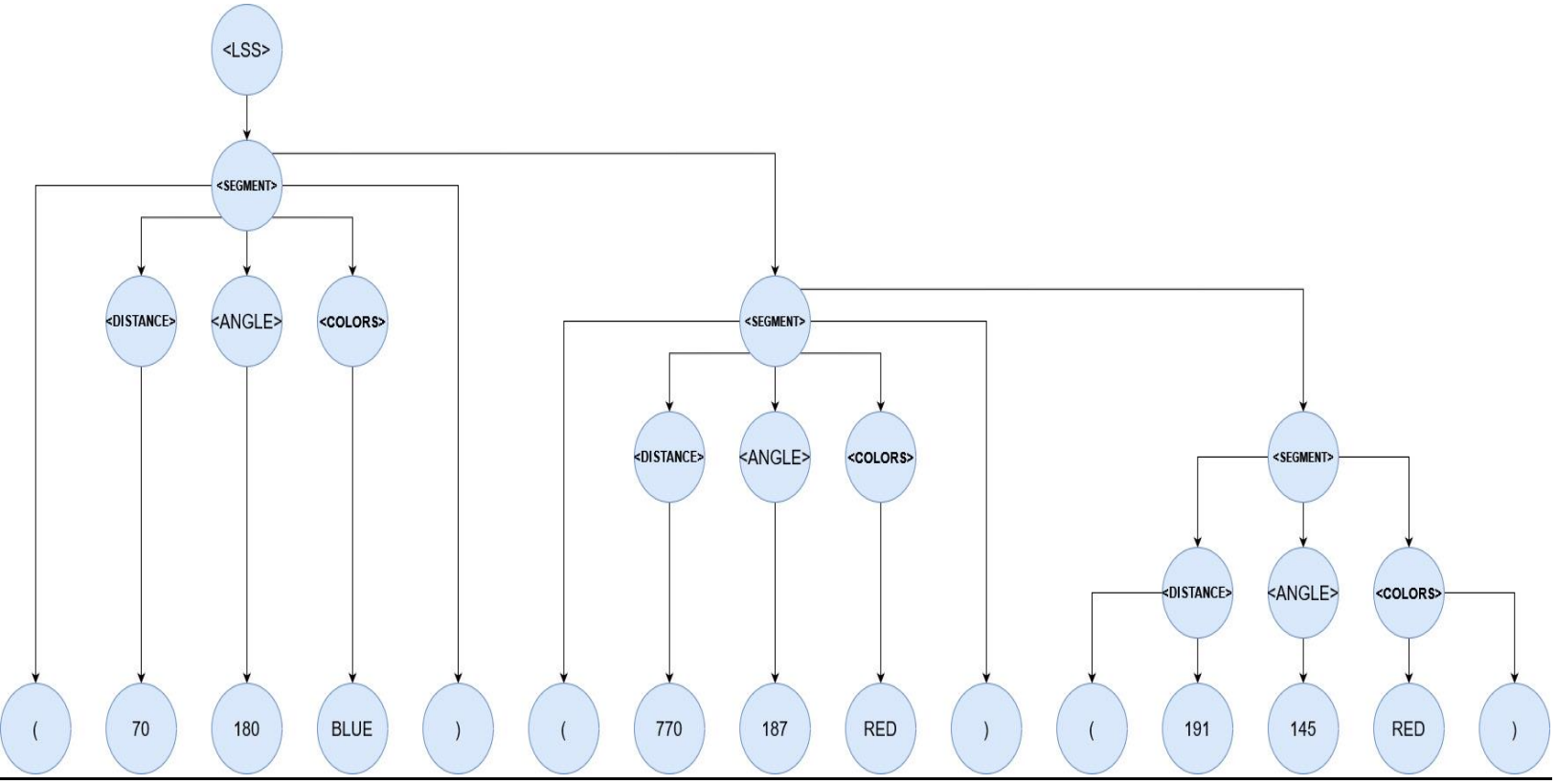
The above rule set will be used to generate the parse trees for following sentences:

1. (120 95 BLACK)
2. (70 180 BLUE) (770 187 RED) (191 145 RED)

Parse Tree 1:



Parse Tree 2:


<LSS>

<DISTANCE>
<ANGLE>
<COLORS>

<DISTANCE>
<ANGLE>
<COLORS>

<DISTANCE>
<ANGLE>
<COLORS>
(70 180 BLUE) (770 187 RED) (191 145 RED)

Problem 5: M-Lines

Consider the language called M-Lines consisting of an event sequence (0 or more events, where an event is the word PLAY or the word REST, or an event sequence sandwiched between RP and LP, or an event sequence sandwiched between LP and RP, or an event sequence sandwiched between S2 and X2, or an event sequence sandwiched between X2 and X2, or an event sequence sandwiched between S3 and X3, or an event sequence sandwiched between S3 and X3.

Examples:

1. PLAY PLAY REST PLAY
2. PLAY PLAY PLAY LP LP LP PLAY RP RP RP RP
3. RP RP PLAY LP PLAY LP X2 PLAY S2 RP RP LP LP

Start: { M_LINES }

Tokens: { PLAY, REST, RP, LP, S2, S3, X2, X3}

Non-Terminals: { <EVENT>,<SEQ> Empty }

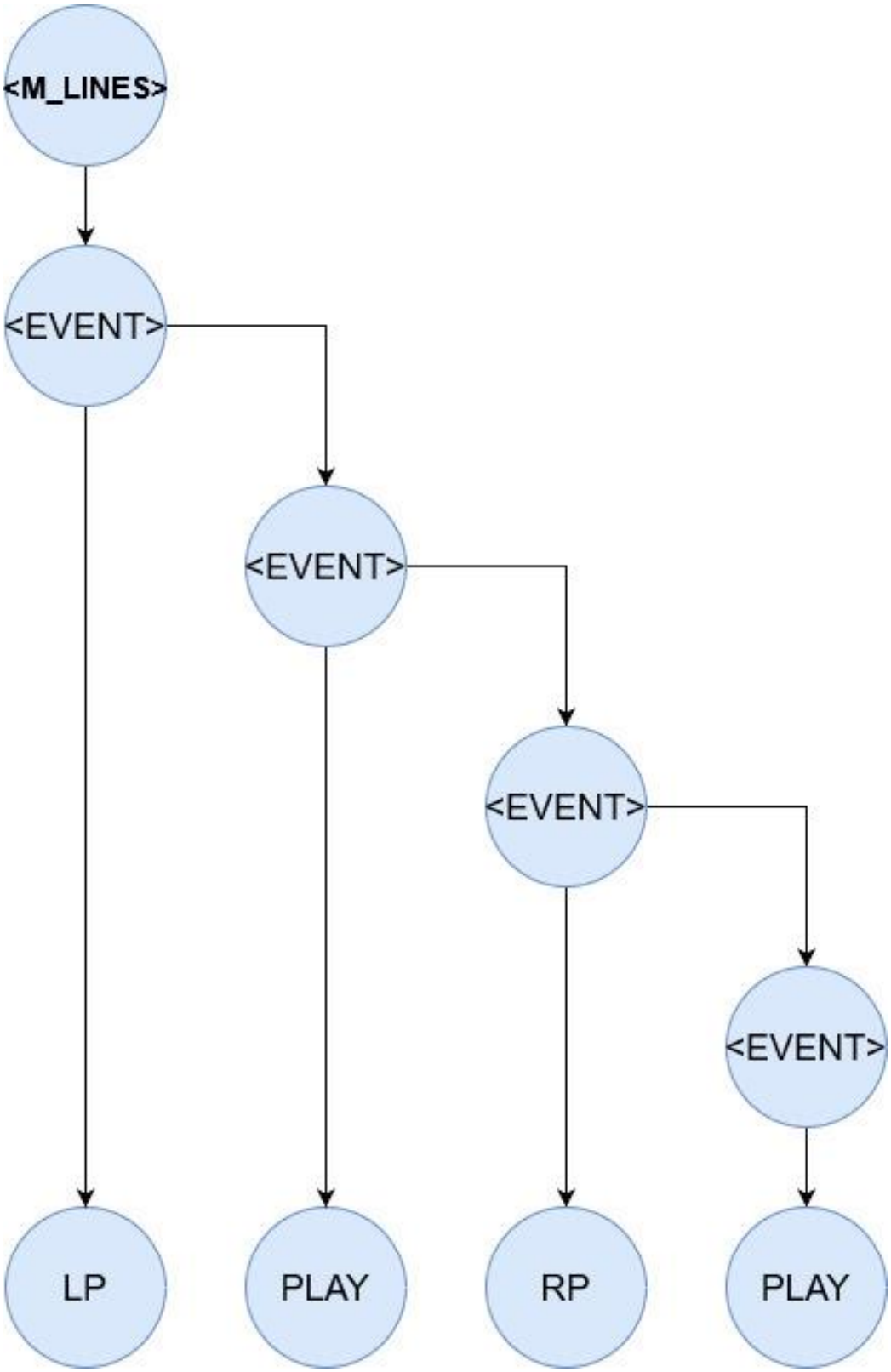
Production Rule Set:

<M_LINES> ::= <EVENT> | <Empty>
<EVENT> ::= PLAY | REST | PLAY <EVENT> | REST <EVENT> |
RP <EVENT> | LP <EVENT> | S2 <EVENT> | S3 <EVENT> |
X2 <EVENT> | X3 <EVENT>| <Empty>

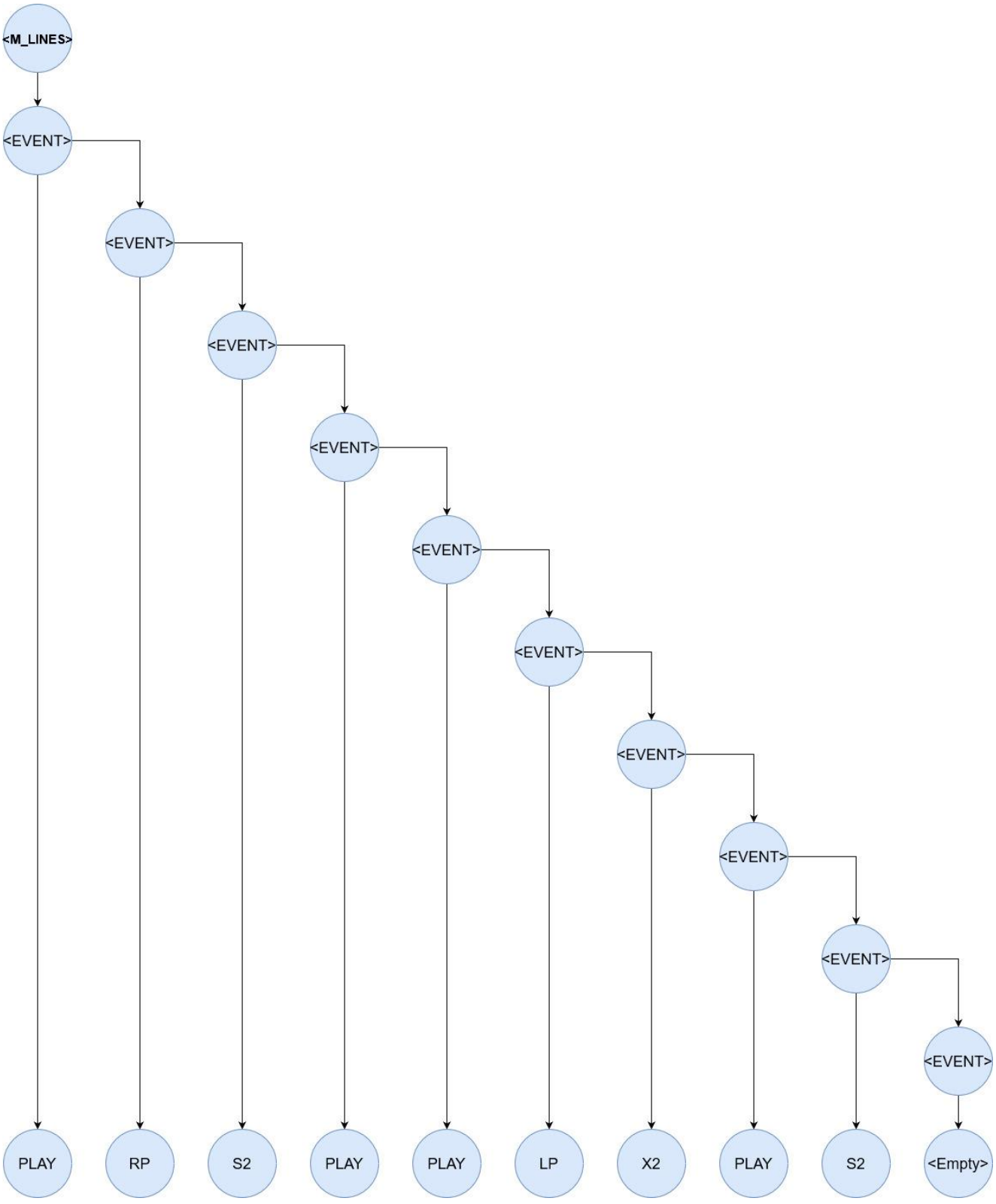
The above rule set will be used to generate the parse trees for following sentences:

1. LP PLAY RP PLAY
2. PLAY RP S2 PLAY PLAY X2 LP X2 PLAY S2

Parse Tree 1:



Parse Tree 2:


<M_LINES>
<EVENT>
<EVENT>
<EVENT>
<EVENT>
<EVENT>
<EVENT>
<EVENT>
<EVENT>
<EVENT>
<EVENT>
PLAY
RP
S2
PLAY
PLAY
LP
X2
PLAY
S2
<Empty>

Problem 6: BNF? (Explanation for a freshman)

BNF or Backus normal form is a language style. In your future courses as a computer science major you will learn and write programs in various types of programming languages. These languages will have specific grammar, syntax, and rules that govern and make up the language. BNF is one form with a specific style that employs a collection of tokens, non-terminal symbols, productions and a start symbol. As you go deeper into your academic career, you will learn the importance of the contents of this collection.