First Problem Set Assignment on BNF

Abstract

This paper focuses exclusively on BNF and features my work creating BNF grammar and building the parse trees connected to each grammar.

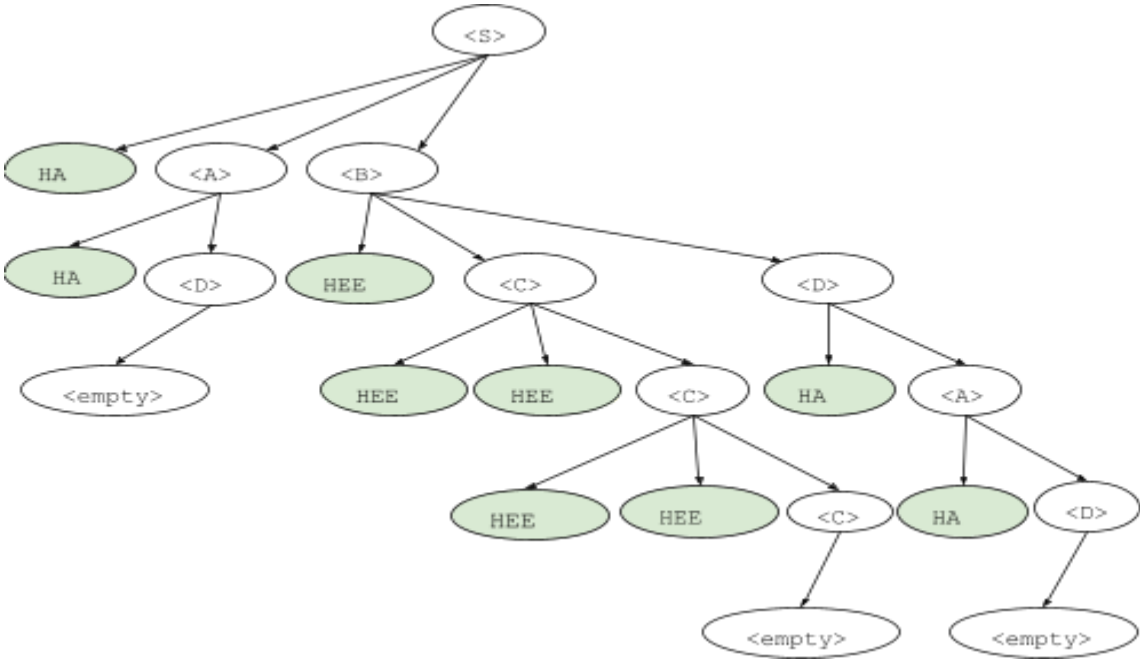
Problem 1 - Laughter

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.

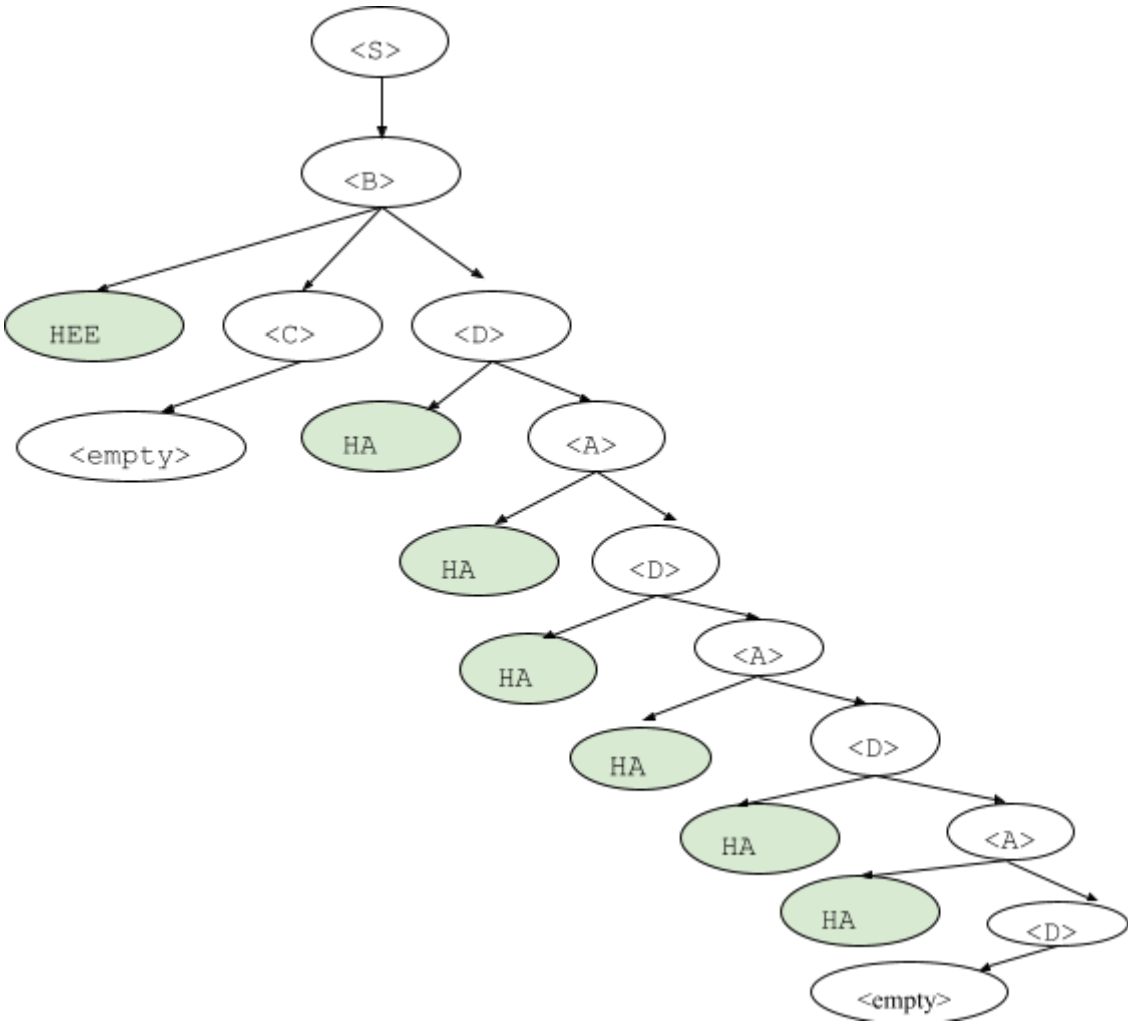
1. BNF grammar:

```
<S> ::= HA <A> <B> | <B>
<A> ::= HA <D>
<B> ::= <empty> | HEE <C> <D>
<C> ::= <empty> | HEE HEE <C>
<D> ::= <empty> | HA <A>
```

2. Parse tree for `HA HA HEE HEE HEE HEE HEE HA HA`



3. Parse tree for `HEE HA HA HA HA HA HA`



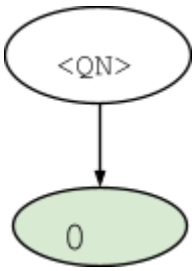
Problem 2 - SQN

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.

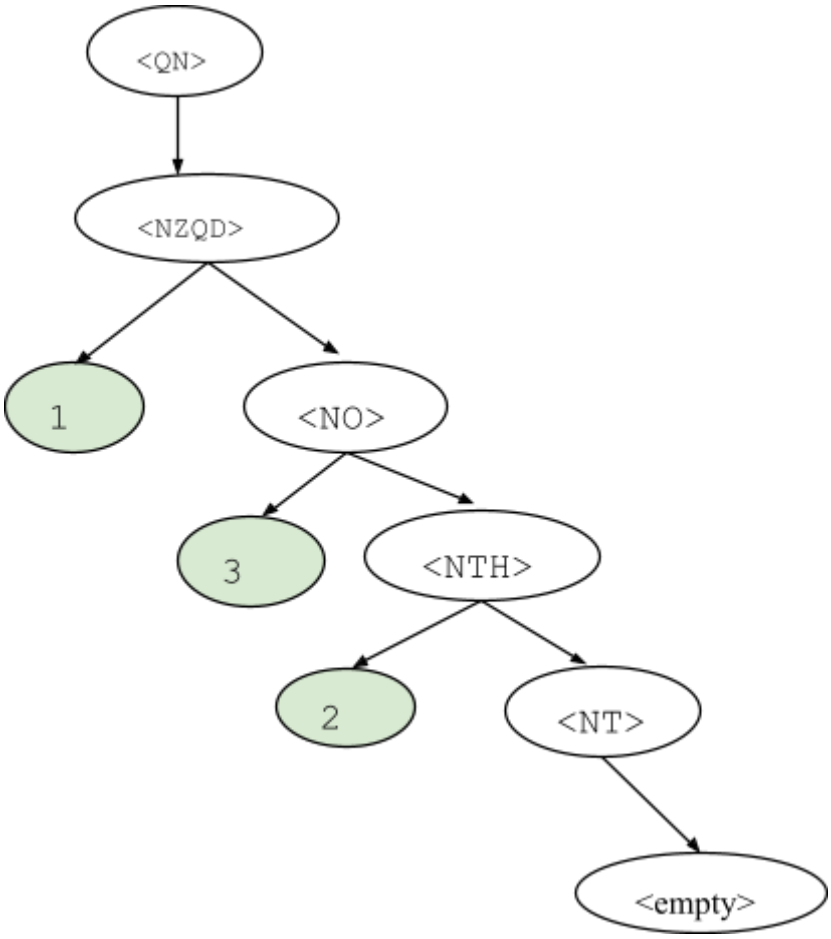
1. BNF grammar:

```
<QN> ::= "0" | <NZQD>
<NZQD> ::= "1" <NO> | "2" <NT> | "3" <NTH>
<A> ::= "1" <NO> | "2" <NT> | "3" <NTH> | " "
<NO> ::= "3" <NTH> | "2" <NT> | "0" <A> | " "
<NT> ::= "3" <NTH> | "1" <NO> | "0" <A> | " "
<NTH> ::= "2" <NT> | "1" <NO> | "0" <A> | " "
```

2. **Parse tree for** `0`**:**



3. **Parse tree for** `132`**:**



4. Explain, in precise terms, why you cannot draw a parse tree consistent with the BNF grammar that you crafted for the string: 1223

Since the grammar for this problem is set up so that two contiguous instances of the same quaternary digit won't be accepted, it is impossible to create the parse tree for the string 1223.

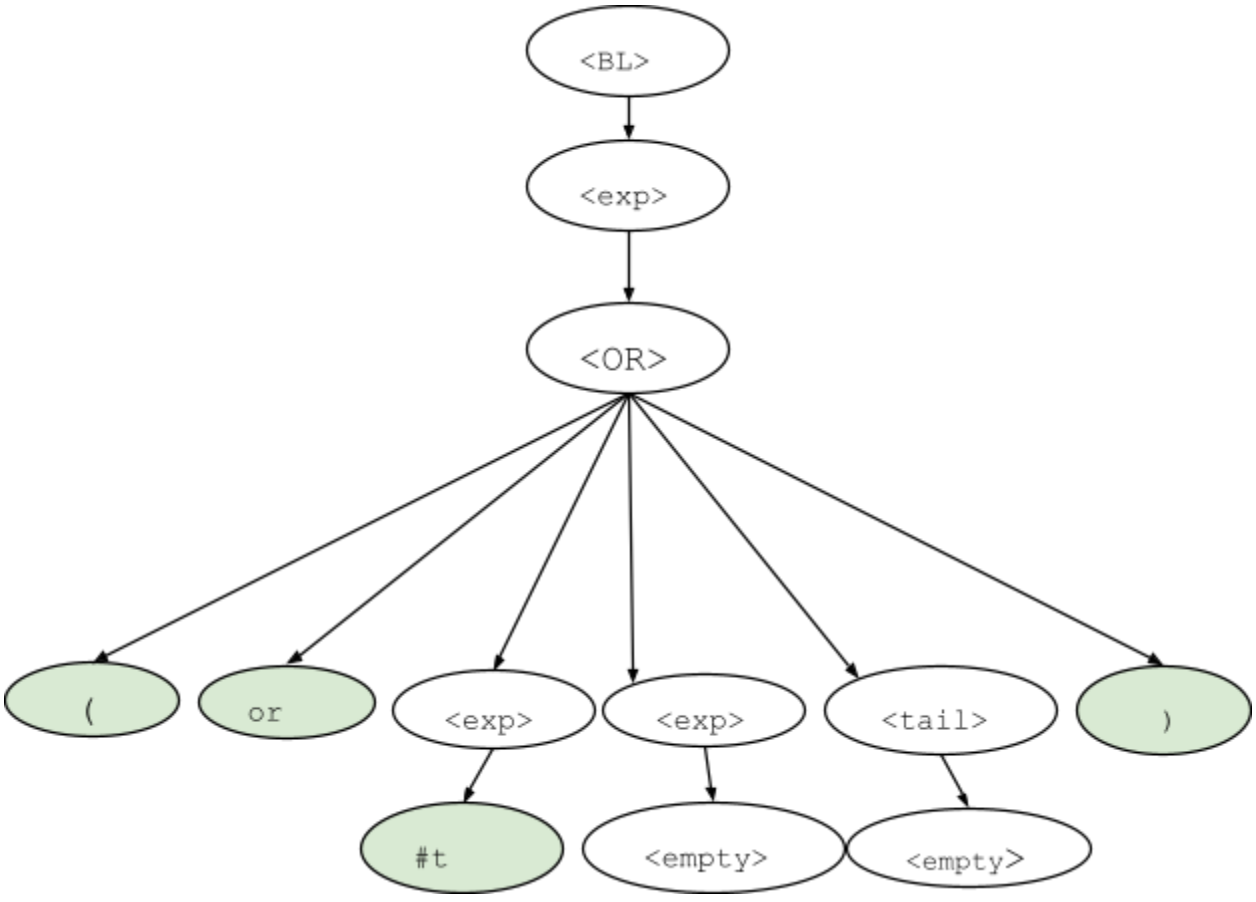
Problem 3 - BXR

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.

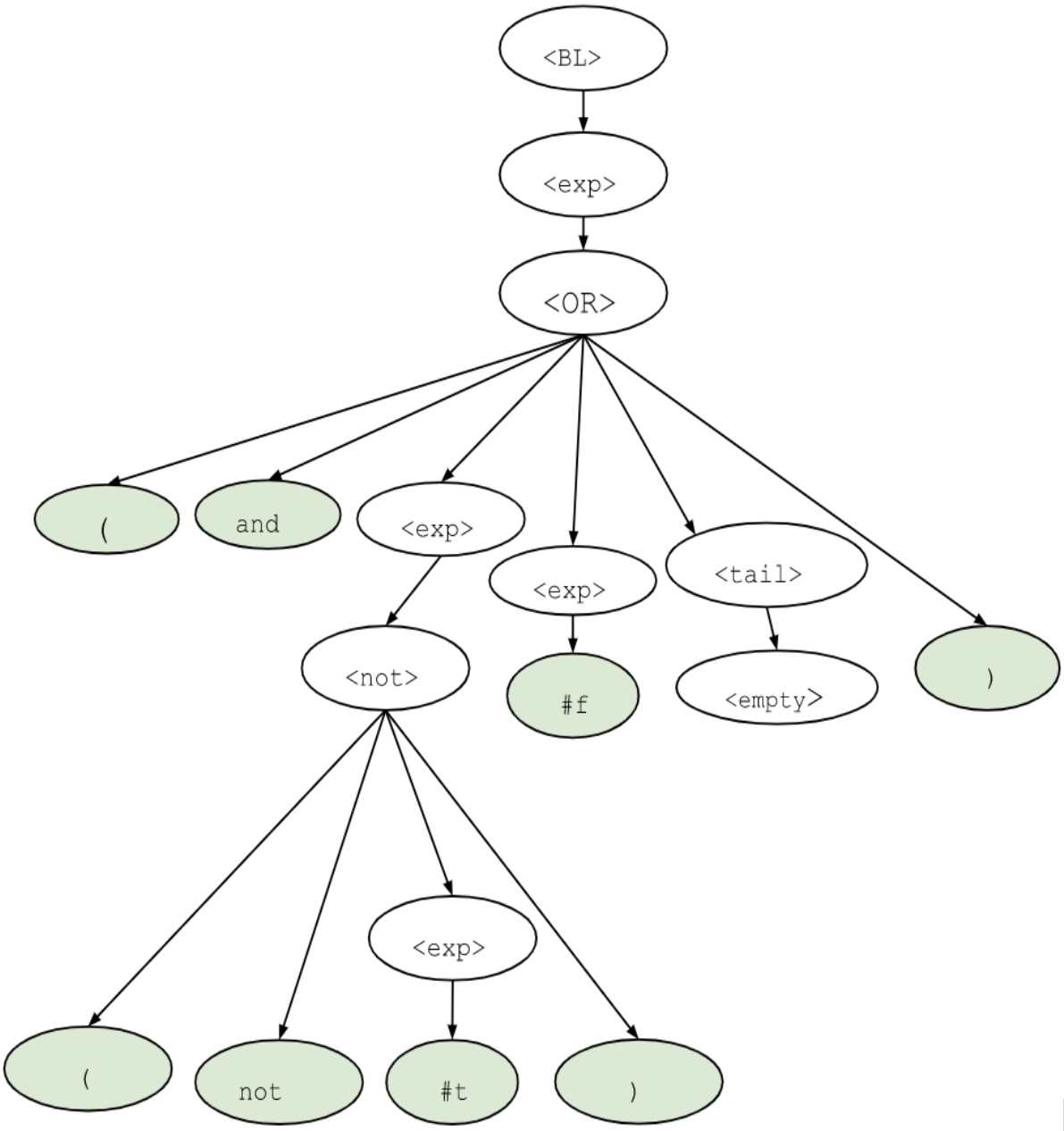
1. BNF grammar:

```
<BL> ::= <exp>
<and> ::= ( and  <exp> <exp> <tail> )
<tail> ::= <empty> | <exp> <tail>
<exp> ::= #t  | #f  | <and> | <or> | <not> | <empty>
<or> ::= ( or  <exp> <exp> <tail> )
<not> ::= ( not  <exp> )
```

2. Parse tree for `( or #t )`



Parse tree for `( and ( not #t ) #f )`



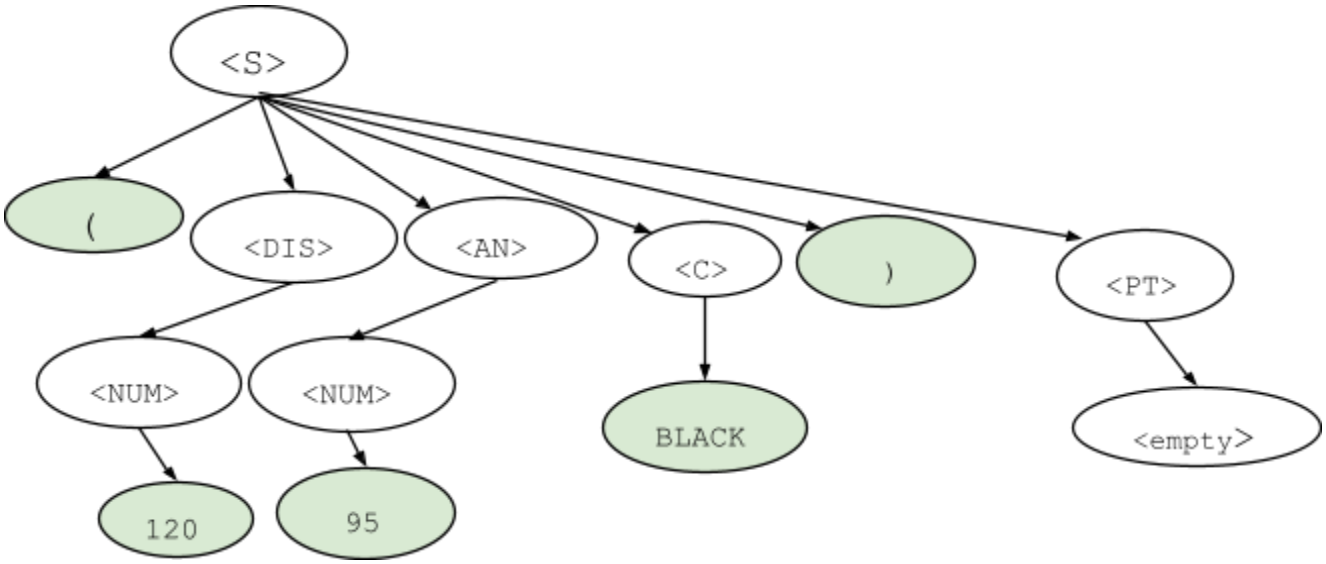
Problem 4 - LSS

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.

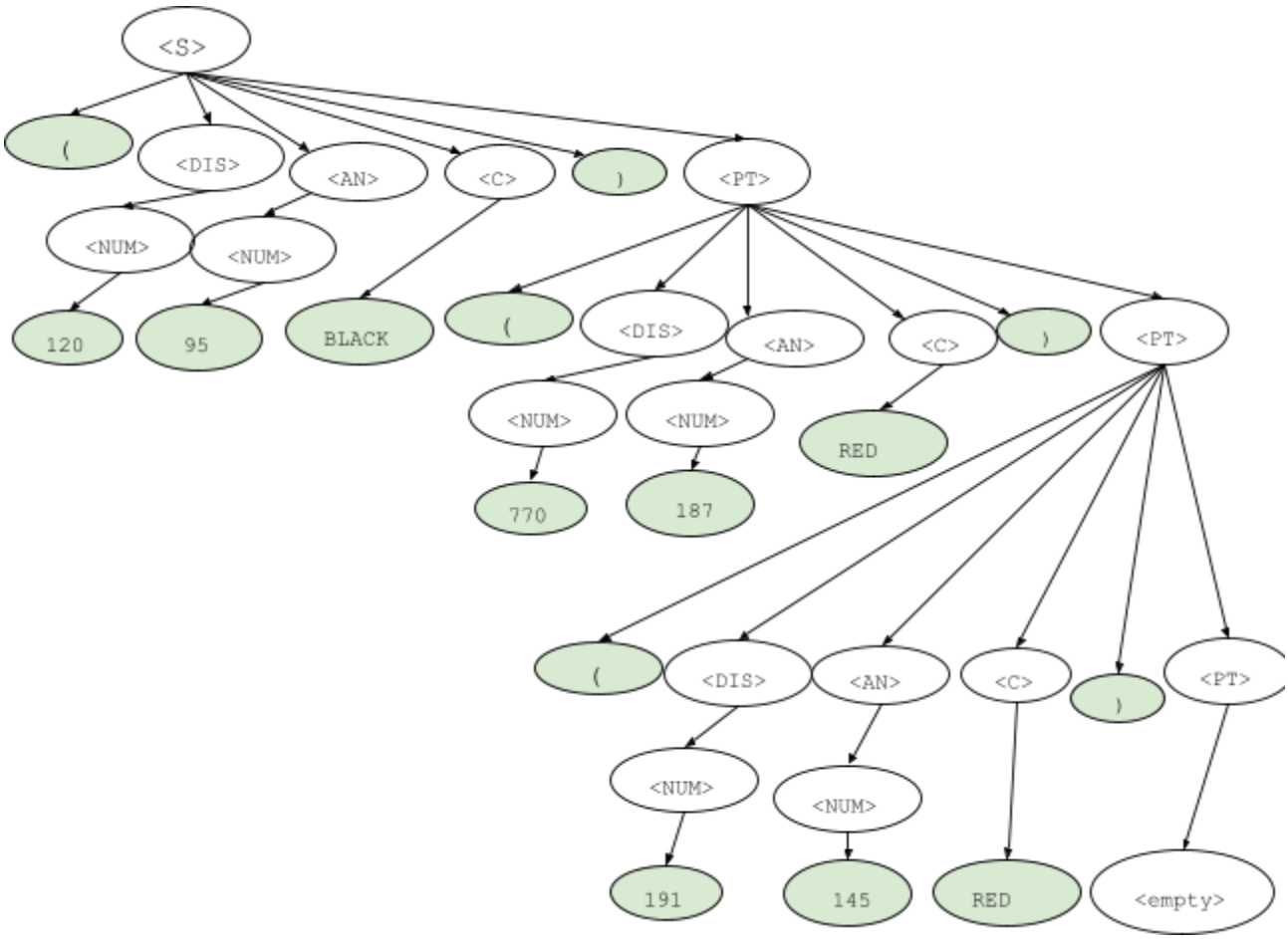
1. BNF grammar:

```
<PT> ::= <empty> | ( <DIS> <AN> <COLORS> ) <PT>
<DIS> ::= <NUM>
<AN> ::= <NUM>
<COLORS> ::= RED | BLACK | BLUE
```

2. Parse tree for `(120 95 BLACK)`**:**



3. Parse tree for `(70 180 BLUE)(770 187 RED)(191 145 RED)`**:**



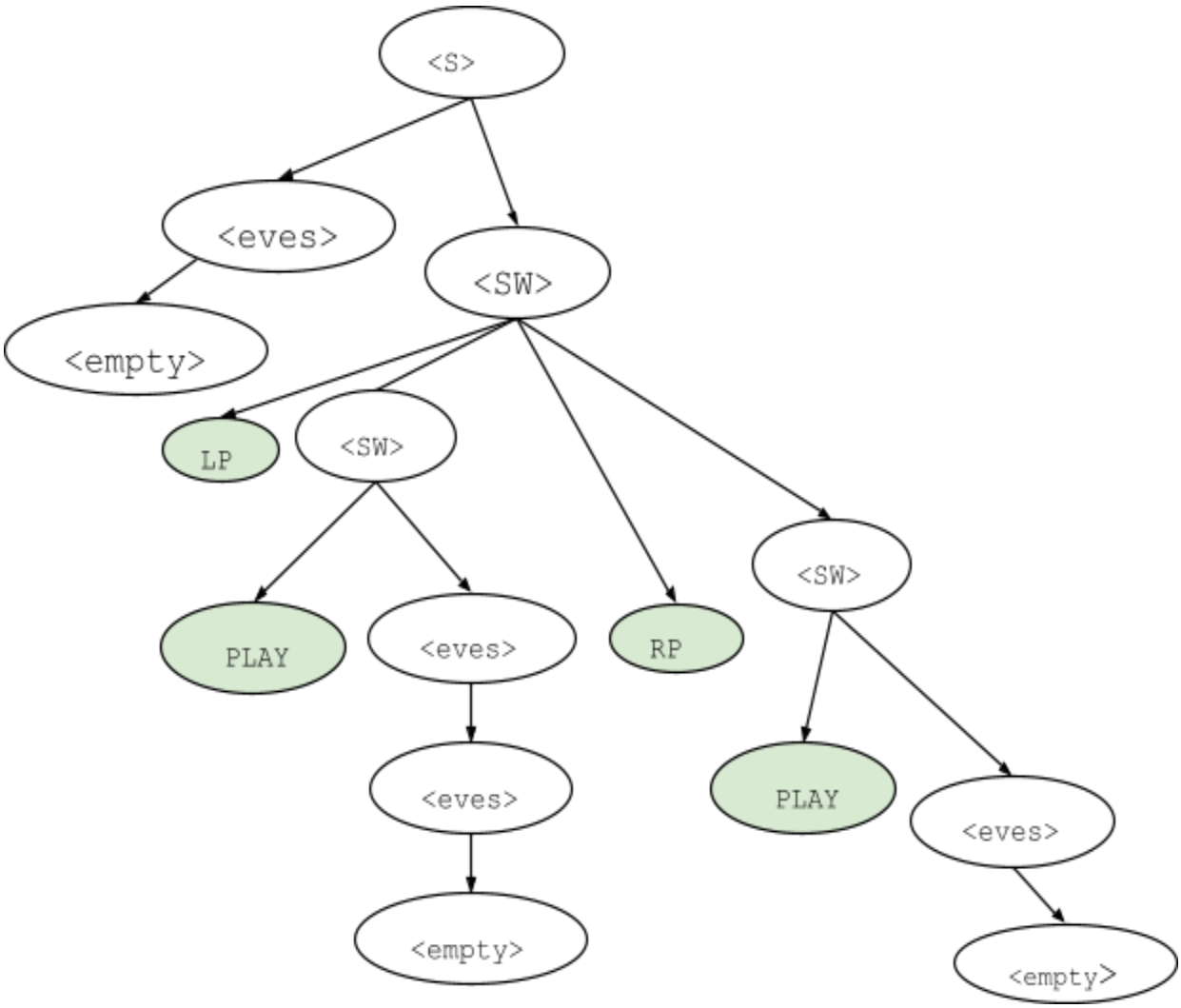
Problem 5 - LSS

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.

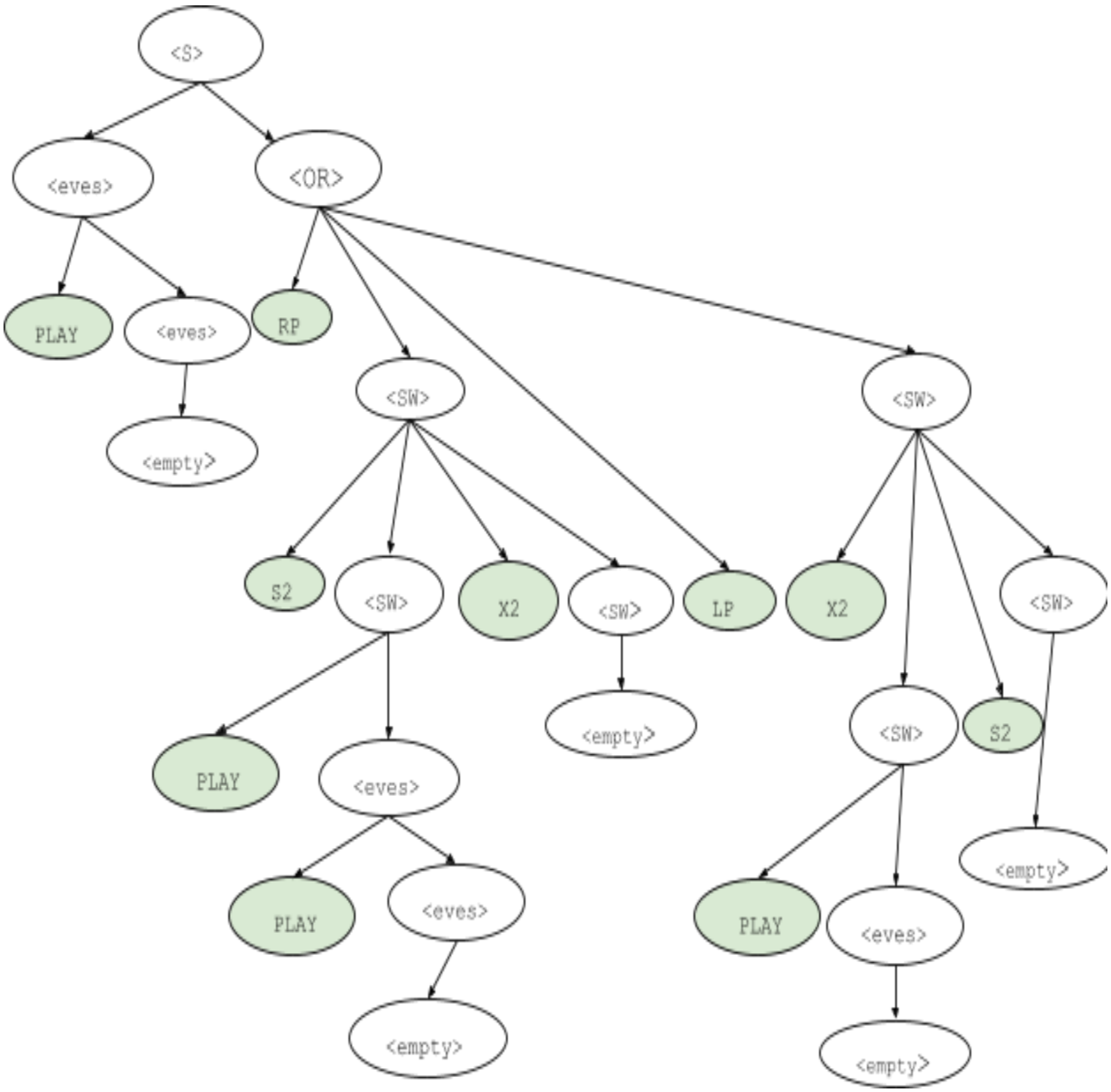
1. BNF grammar:

```
<S>    ::= <eve> | <eves> <sw>
<eve>  ::= PLAY <eves> | REST <eves>
<eves> ::= PLAY <eves> | REST <eves> | <sw> | <empty>
<SW>   ::= PLAY <eves> | REST <eves> | RP <SW> LP <SW>
         | LP <SW> RP <SW> | S2 <SW> X2 <SW> | X2 <SW> S2 <SW>
         | S3 <SW> X3 <SW> | X3 <SW> S3 <SW> | <empty>
```

1. Parse tree for `LP PLAY RP PLAY`



2. Parse tree for `PLAY RP S2 PLAY PLAY X2 LP X2 PLAY S2`



Problem 6 - BNF?

Imagine that a freshman computer science major asks you the question: "What is BNF?" Please write an answer, in natural language (English, please), without examples, in a manner that you believe will serve to meaningfully inform the student about the nature and significance of BNF.

Backus-Naur notation, or BNF for short, is a grammar-writing system used to specify computer languages. In a BNF grammar, there are four entities:

1. **Tokens** that are regarded as being a component of the language under the definition
2. **Symbols** that aren't actually part of the language being described yet are crucial to its definition
3. **Productions**, often known as rewriting rules, that can be used to convert a nonterminal symbol into a string of tokens and nonterminals
4. **Start symbol**, a nonterminal symbol that conceptually exemplifies the language being described, is used.