Sven Kappeler - BNF Assignment

Learning Abstract

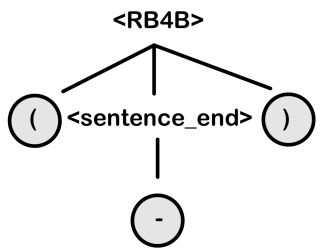
This assignment is an introduction to BNF. The purpose of this assignment is to perform some examples of making a BNF Language and parse trees with said language.

Problem 1

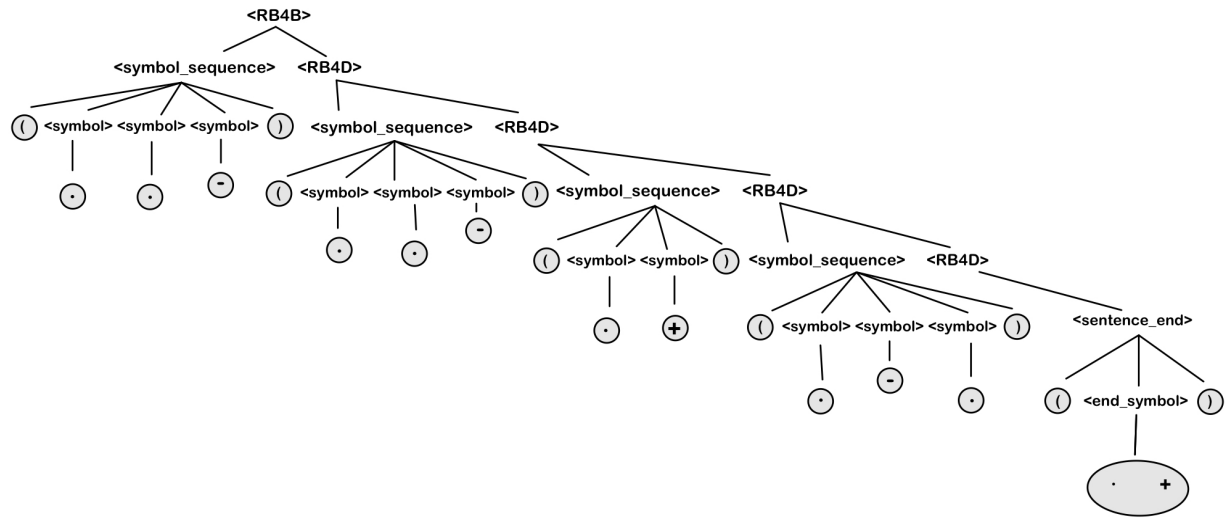
1)

```
<RB4B> ::= <symbol_sequence>   <RB4B> | <sentence_end>
<symbol_sequence> ::= ( <symbol> ) | ( <symbol> <symbol> ) | ( <symbol> <symbol> <symbol> )
<sentence_end> ::= ( <end_symbol> )
<end_symbol> ::=  -  |  .  +
<symbol> ::=  -  |  .  |
```

2)



3)

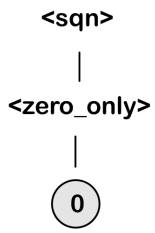


Problem 2

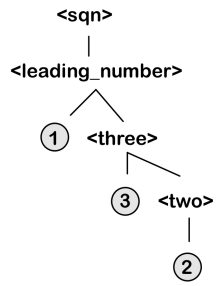
1)

```
<sqn> ::=  <zero_only> | <leading_number>
<zero_only> ::= "0"
<leading_number> ::= <one> | <two> | <three>
<zero> ::= "0" | "0" <one> | "0" <two> | "0" <three>
<one> ::= "1" | "1" <zero> | "1" <two> | "1" <three>
<two> ::= "2" | "2" <zero> | "2" <one> | "2" <three>
<three> ::= "3" | "3" <zero> | "3" <one> | "3" <two>
```

2)



3)



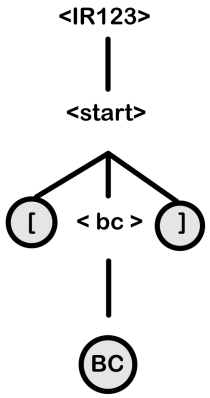
4) The terminal “2” can never be followed directly by another terminal “2” because according to my BNF <two> contains a terminal “2” and then a different number’s nonterminal; this different number’s nonterminal starts with either 0 | 1 | 3. It does not point to another <two>, thus the next number can never be repeated.

Problem 3

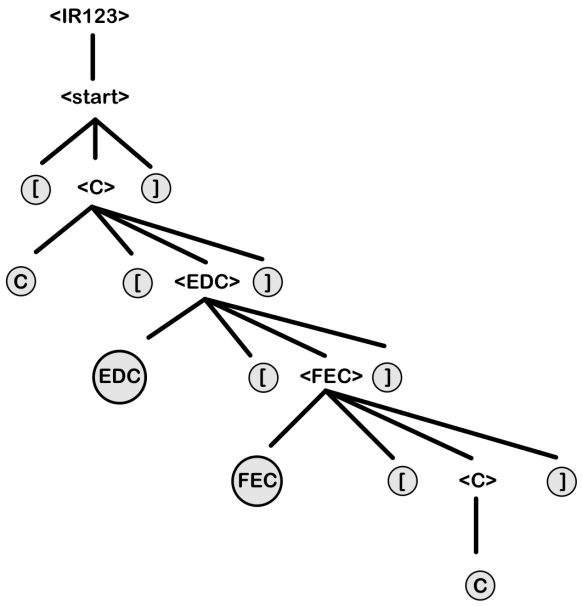
1)

```
<IR123> ::= <start>
<start> ::= [ < C > ] | [ < DC > ] | [ < BC > ] | [ < EDC > ] | [ < FEC > ] | [ < GFC > ]
< C > ::=  C |  C [ < DC > ] |  C [ < BC > ] |  C [ < EDC > ] |  C [ < FEC > ] |  C [ < GFC > ]
< DC > ::=  D C |  D C [ < C > ] |  D C [ < BC > ] |  D C [ < EDC > ] |  D C [ < FEC > ] |  D C [ < GFC > ]
< BC > ::=  B C |  B C [ < C > ] |  B C [ < DC > ] |  B C [ < EDC > ] |  B C [ < FEC > ] |  B C [ < GFC > ]
< EDC > ::=  E D C |  E D C [ < C > ] | E D C [ < DC > ] | E D C [ < BC > ] | E D C [ < FEC > ] | E D C [ < GFC > ]
< FEC > ::= F E C | F E C [ < C > ] | F E C [ < DC > ] | F E C [ < BC > ] | F E C [ < EDC > ] | F E C [ < GFC > ]
< GFC > ::= G F C | G F C [ < C > ] | G F C [ < DC > ] | G F C [ < BC > ] | G F C [ < EDC > ] | G F C [ < FEC > ]
```

2)



3)



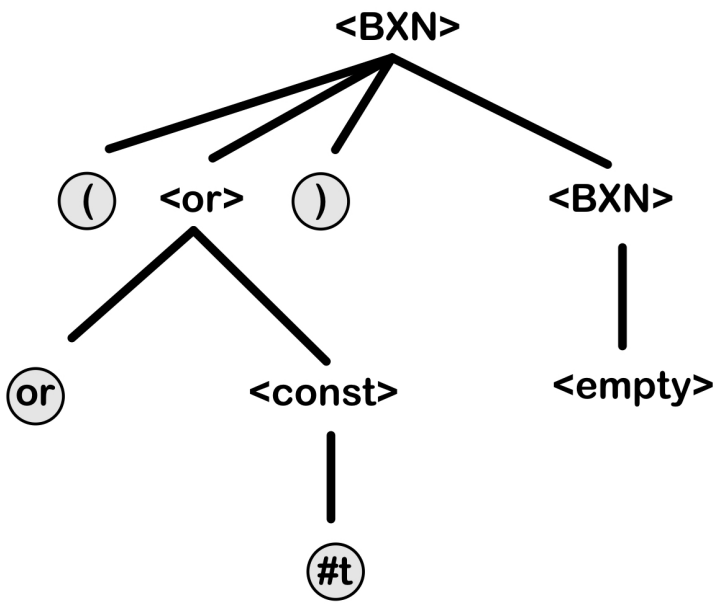
4)
The nonterminal <BC> can never have another nonterminal <BC> trailing it. <BC> never points directly to another <BC>; it points to all the other nonterminals.

Problem 4

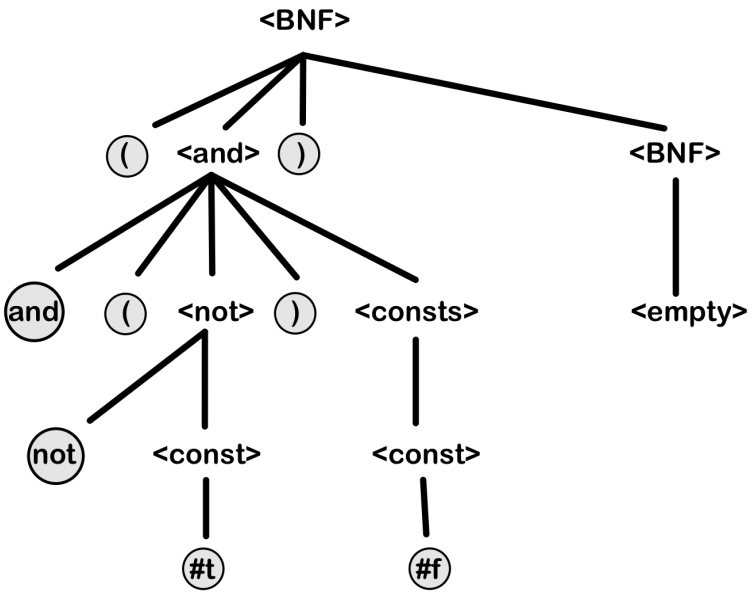
1)

```
<BXN> ::= ( <and> ) <BXN> | ( <or> ) <BXN> | <empty>
<and> ::= "and" | "and" <consts> | "and" <const> ( <not> ) | "and" ( <not> ) <consts>
<or> ::= "or" | "or" <consts> | "or" ( <not> ) <consts>
<consts> ::= <const> | <const> <consts>
<not> ::= "not" <const>
<const> ::= "#t" | "#f"
```

2)



3)

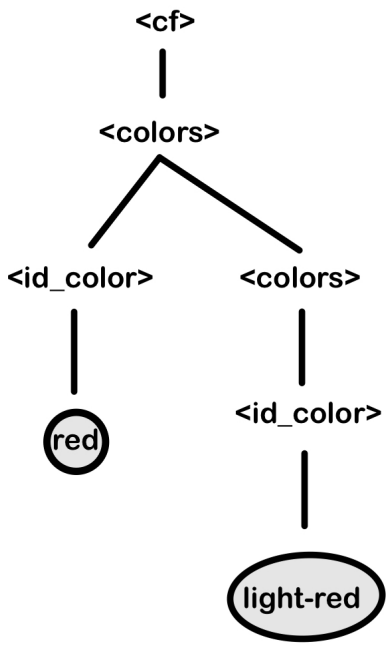


Problem 5

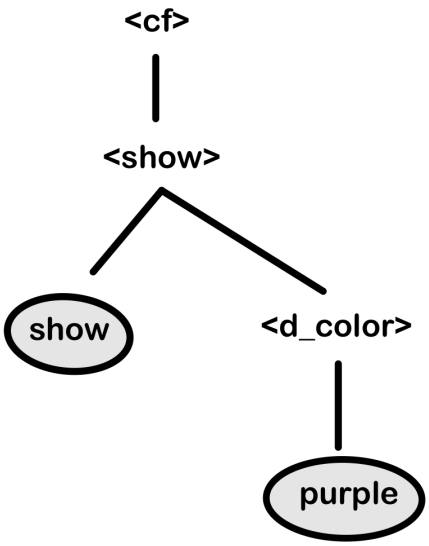
1)

```
<cf> ::= <show> | <describe> | <add> | <colors>
<colors> ::= <id_color> | <id_color> <colors>
<id_color> ::= "color_name"
<show> ::= "show" <d_color>
<d_color> ::= color
<describe> ::= ( <n_color> )
<n_color> ::= <digit> <digit> <digit> | <digit> <digit> <digit> <opacity>
<digit> ::= "0" | "1" | "2" | "3" | ... | "255"
<opacity> ::= "1" | "2" | "3" | ... | "255"
<add> ::= "add" ( <n_color> ) <id_color>
```

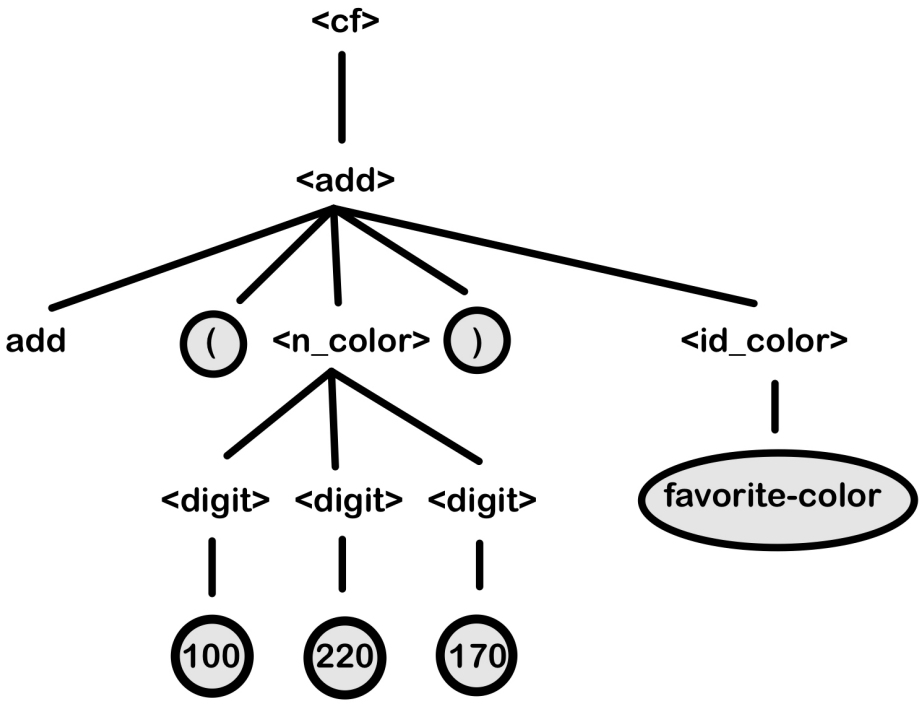
2)


<cf>
<colors>
<id_color>
<colors>
red
<id_color>
light-red

3)


<cf>
<show>
show
<d_color>
purple

4)



Problem 6

BNF is a useful way to describe the syntax of a language; that is the rules of how the language can be formulated. Using a finite of rules it is possible to describe an infinite amount of sentences or parse trees that can be formed from said rules. These rules consist of two distinct items: nonterminals symbols and terminal tokens. Terminal tokens are the very bottom of the parse tree, these include numbers, letters, words, and most characters: ( ) [ ] “_”. The nonterminals are surrounded by “< >” and is essentially a rule describing which and what the latter nonterminals or tokens including how they can be used. At the very top of the parse tree is the start symbol.