BNF Assignment: First Interactions

Learning Abstract:

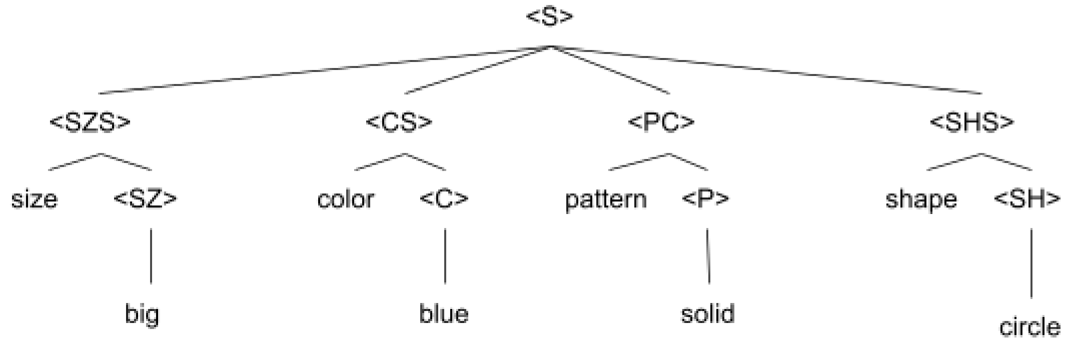
This is a project dealing with BNF (Backus-Naur Forms or Backus Normal Forms) and how they are built. These projects will mostly be small different languages defined using BNF grammar descriptions. There will be parse trees and explanations for membership to a language.

Language #1: Shapes

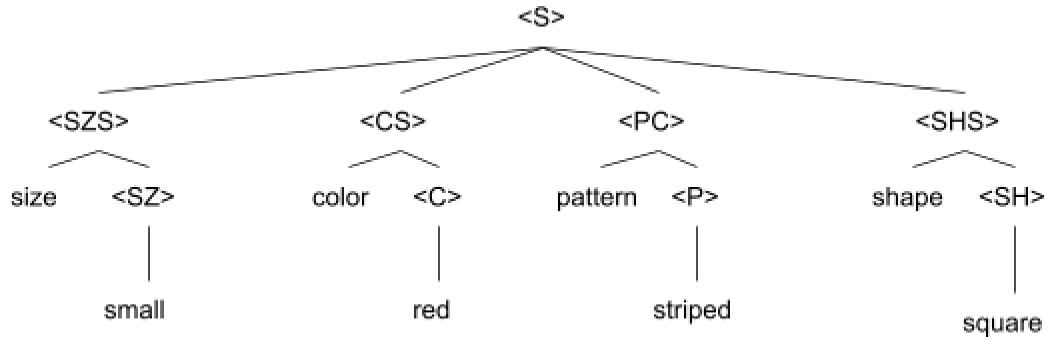
Definition:

1. **Tokens**: { size, color, pattern, shape, big, medium, small, red, blue, yellow, striped, dotted, solid, square, triangle, circle }
2. **Non-Terminals**: { S, SZS, SZ, SC, C, PS, P, SHS, SH }
3. **Productions**:
   <S> : := (<SZS> <CS> <PS> <SHS>)
   <SZS> : := (size <SZ>)
   <SZ> : := big | medium | small
   <CS> : := (color <C>)
   <C> : := red | blue | yellow
   <PS> : := (pattern <P>)
   <P> : := striped | dotted | solid
   <SHS> : := (shape <SH>)
   <SH> : := square | triangle | circle
4. **Start Symbol**: S

Parse Tree: ((size big) (color blue) (pattern solid) (shape circle))



Parse Tree: ((size small) (color red) (pattern striped) (shape square))

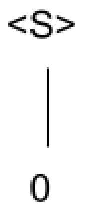


Language #2: Special Quaternary Numbers (SQN)

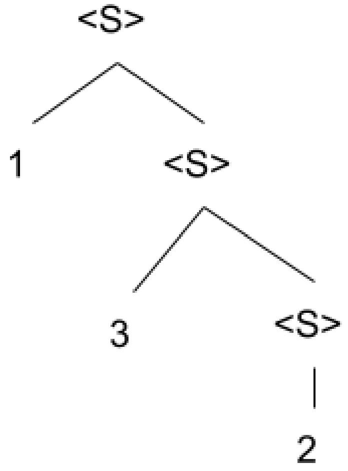
Definition:

1. **Tokens**: { 0, 1, 2, 3 }
2. **Non-Terminals**: { S }
3. **Productions**:
   <S> : : = 1<S>
   <S> : : = 2<S>
   <S> : : = 3<S>
   <S> : : = <S>0<S>
   <S> : : = 0 | 1 | 2 | 3
4. **Start Symbol**: S

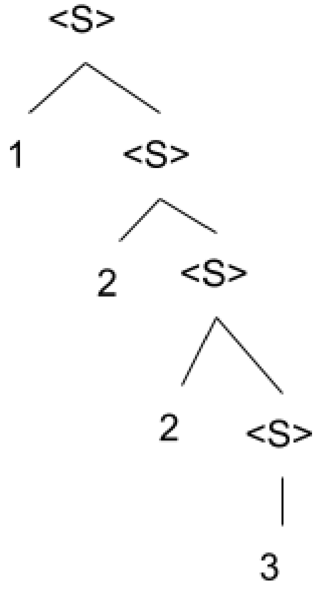
Parse Tree: 0



Parse Tree: 132



Parse Tree: 1223



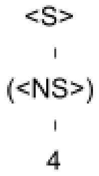
Language #3: Fours

Definition:

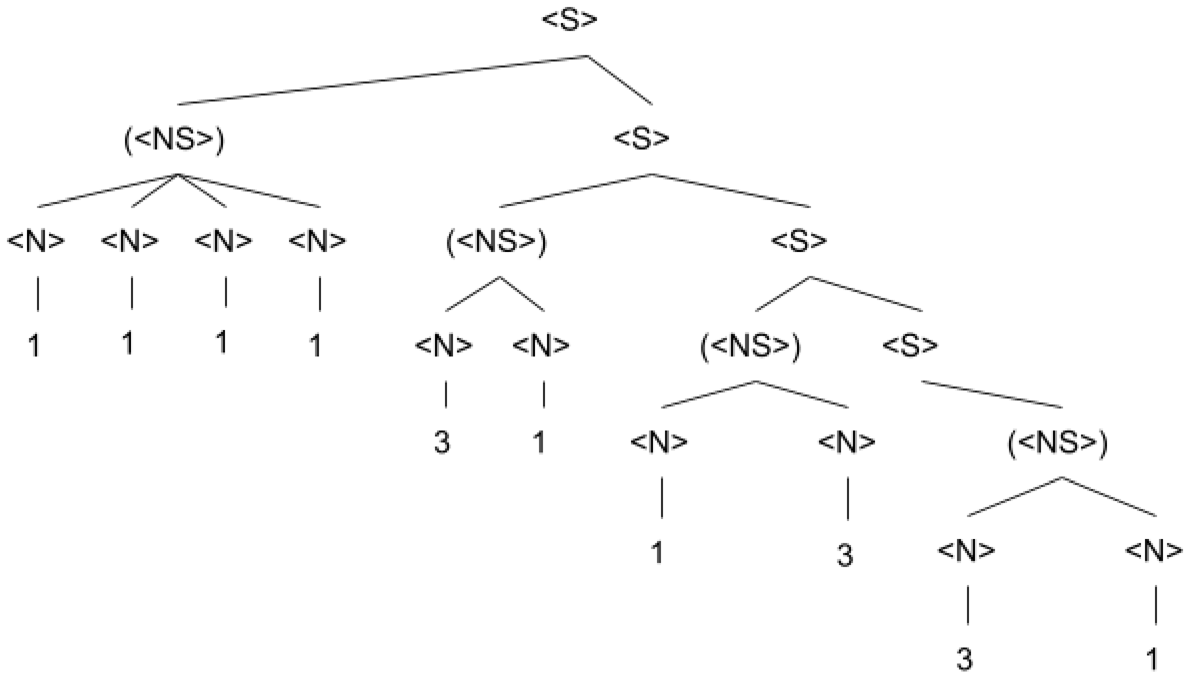
5. **Tokens**: { 1, 2, 3, 4 }
6. **Non-Terminals**: { S, NS, N }
7. **Producions**:
   <S> : := (<NS>) | (<NS>) <S>
   <NS> : := 4 | <N><N> | <N><N><N> | <N><N><N><N>
   <NS> : := Σ <N> = 4
   <N> : := 1 | 2 | 3

8. **Start Symbol**: S

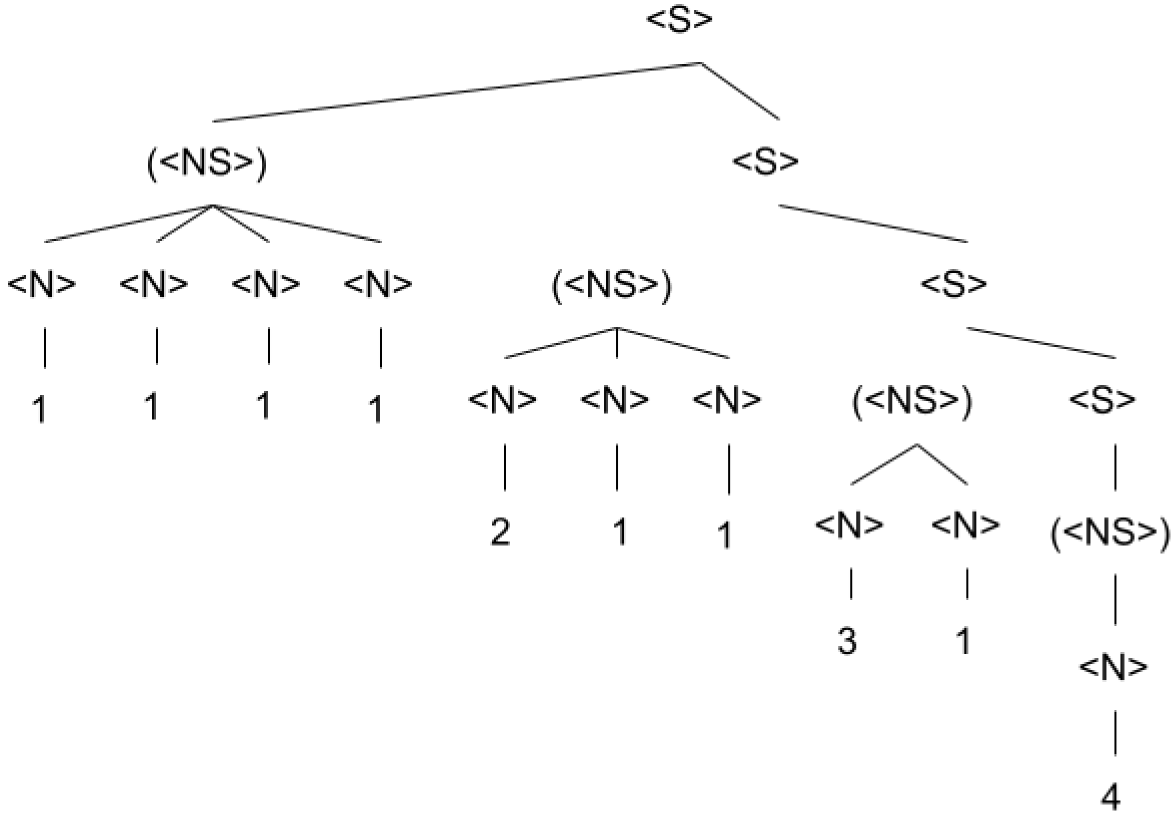
Parse Tree: (4)



Parse Tree: (1 1 1 1) (3 1) (1 3) (3 1)



Parse Tree: (1 1 1 1) (2 1 1) (3 1) (4)

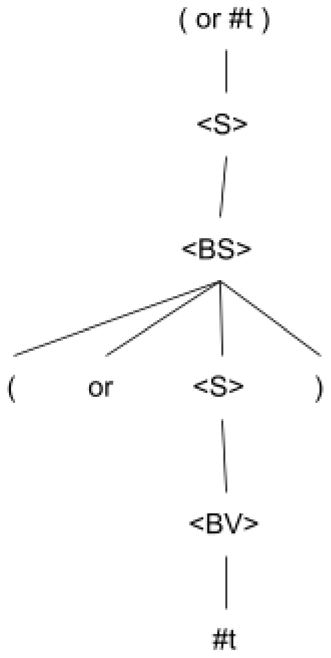


Language #4: BXR (Boolean Expressions in Racket)

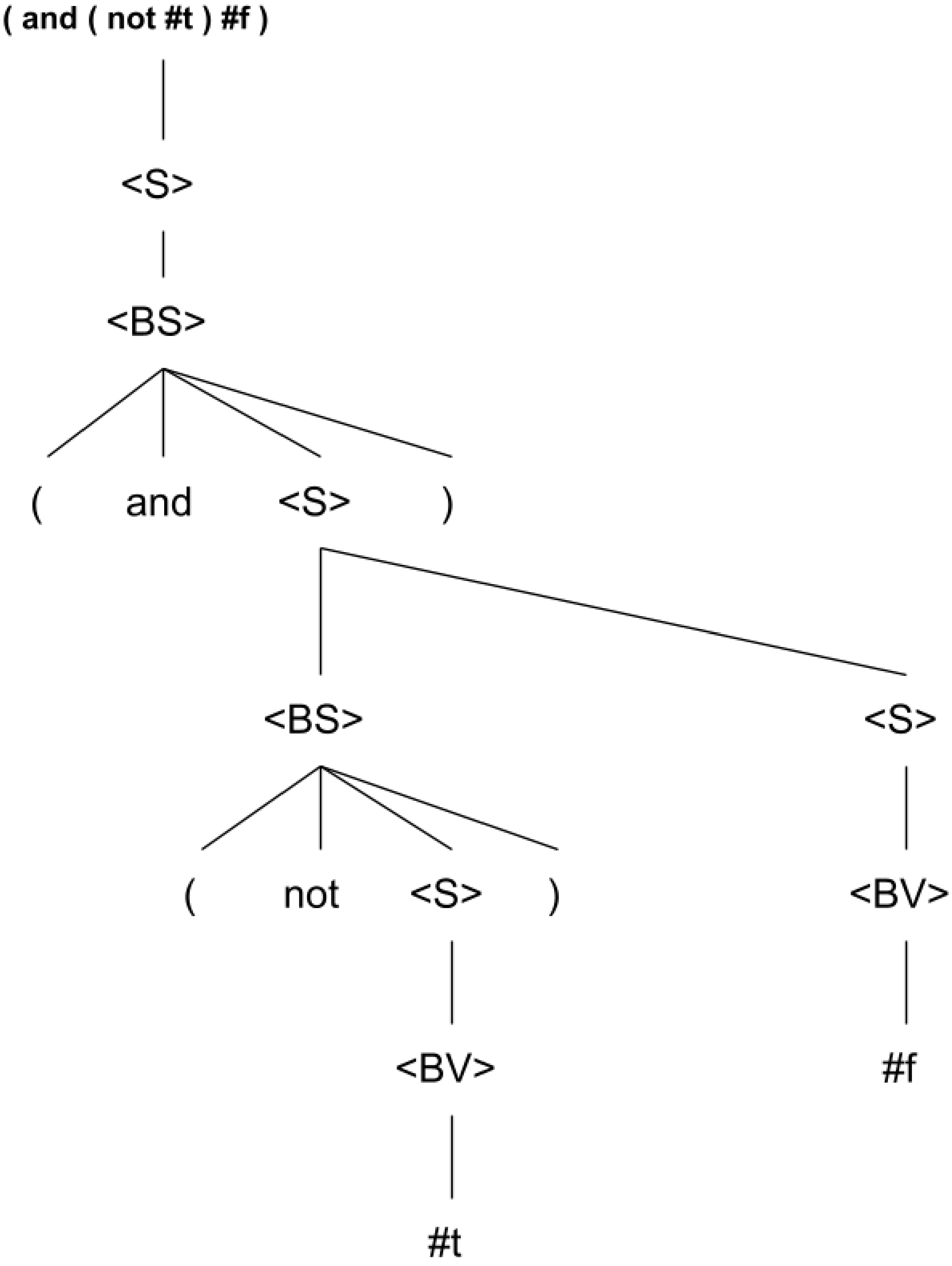
Definition:

1. **Tokens**: { not, or, and, #t, #f }
2. **Non-Terminals**: { S, BS, BV }
3. **Producions**:
   <S> : := <BS> | <BV> | <BS> <S> | <BV> <S> | <empty>
   <BS> : := (and <S>) | (or <S>) | (not <S>)
   <BV> : := #t, #f
4. **Start Symbol**: S

Parse Tree: (or #t)



Parse Tree: (and (not #t) #f)

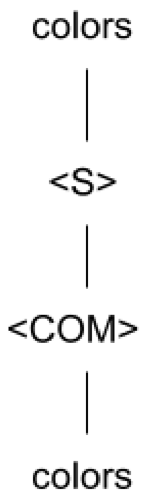


Language #5: Color Fun (CF)

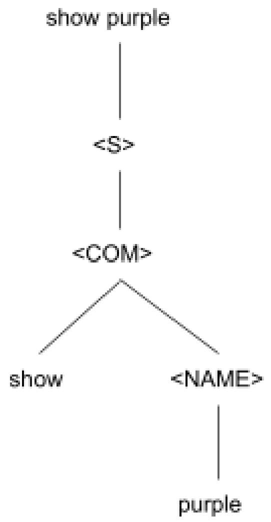
Definition:

1. **Tokens**: { add, color, colors, show, describe, exit, { 0-255 }, { a-z }, { A-Z } }
2. **Non-Terminals**: { S, COM, COLOR, NUM, NAME }
3. **Producions**:
   <S> : := <COM>
   <COM> : := add <COLOR> <NAME> | show <NAME> | describe <NAME> | colors | exit
   <COLOR> : := (<NUM> <NUM> <NUM>) | (<NUM> <NUM> <NUM> <NUM>) | color
   <NUM> : := (0-255)
   <NAME> : := character string comprised of any combination of (0-255) (a-z) (A-Z)
4. **Start Symbol**: S

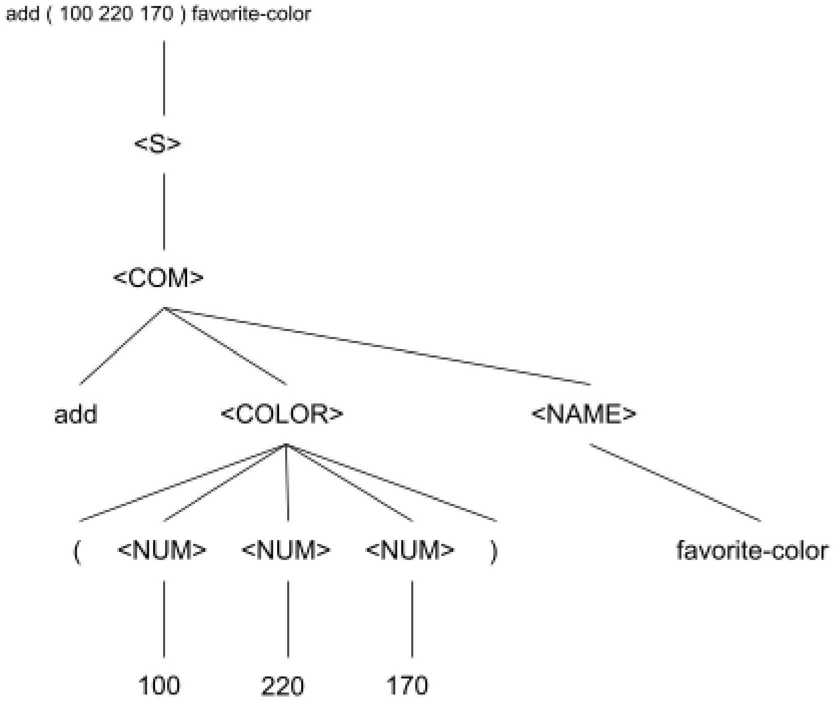
Parse Tree: colors



Parse Tree: show purple



Parse Tree: add (100 220 170) favorite-color



WTF is BNF!? (100 words or less)

BNF is a specific way of describing a language. This could be a spoken language, programming language, or potentially a coded message. The language is described using generic tokens, broken out into their most generic terms. There are several portions; tokens, non-terminal tokens, productions, and a start symbol. The tokens are a set of valid characters or character strings. Non-terminal tokens are the generic forms of the tokens including the startin symbol. Non-terminals **ARE NOT** the same as the tokens themselves. Productions are a description of how the non-terminals are mapped back to the tokens. Start symbol is almost always ‘S’.