Context Free Grammars (process)

We think of context free grammars as **generating** strings.

1. Start from the start symbol S.
2. Choose a nonterminal in the string, and a production rule $A \to w_1|w_2|\ldots|w_k$ replace that copy of the nonterminal with w_i.
3. If no nonterminals remain, you're done! Otherwise, goto step 2.

A string is in the language of the CFG iff it can be generated starting from S.

Examples

$S \to 0S0|1S1|0|1|\varepsilon$

$S \to 0S|S1|\varepsilon$

$S \to (S)|SS|\varepsilon$

The alphabet here is {(,)} i.e. parentheses are the characters.

$S \to AB$

$A \to 0A1|\varepsilon$

$B \to 1B0|\varepsilon$

Arithmetic CFG

$E \to E + E | E * E | (E) | x | y | z | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$

Generate $(2 * x) + y$

Generate $2 + 3 * 4$ in two different ways

How do we encode order of operations

If we want to keep "in order" we want there to be only one possible parse tree.

Differentiate between "things to add" and "things to multiply"

Only introduce a * sign after you've eliminated the possibility of introducing another + sign in that area.

$E \to T | E + T$

$T \to F | T * F$

$F \to (E) | N$

$N \to x | y | z | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$