Removing grammar ambiguity

1. Left recursion.
2. Left factoring.

Both problems prevents any LL parser from deciding deterministically which rule should be fired.

Left Recursion

- We have to eliminate left recursion because top down parsing methods can not handle left recursive grammars.
- A grammar is left recursive if it has a nonterminal A such that there is a derivation A → Aα for some string α
- Consider the left recursive grammar
  A → A α1 | A α2 | β1 | β2

Solution:

A → β1A′| β2A′
A′ → α1A′|α2A′| ε

Left Factoring

- In left factoring it is not clear which of two alternative productions to use to expand a nonterminal A.

  i.e. if A → αβ1 | αβ2

- W e don't know whether to expand A to αβ1 or to αβ2
- To remove left factoring for this grammar replace all A productions containing α as prefix by A → αA′ then A′ → β1 | β2

Example 1:

X → X + X | X * X | D
D → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Remove Left factoring and left recursion.

Solution:

First removing left factoring.
X → X B | D
B → + X | * X
D → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Still the grammar is ambiguous it has left recursion.
X → D X'
X' → B X' | ϵ
B → + X | * X
D → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

A Recursive-Descent Parser

• One parse method per non-terminal symbol

• A non-terminal symbol on the right-hand side of a rewrite rule leads to a call to the parse method for that non-terminal

• A terminal symbol on the right-hand side of a rewrite rule leads to “consuming” that token from the input token string

• | in the CFG leads to “if-else” in the parser

• ()* in the CFG leads to “while” in the parser

Task:

Do a small parser program with the previous grammar (Example 1) after removing its ambiguity.

1- Remove left factoring and left recursion.
2- Toknoizer
3- Prepare node classes to build parse tree.
4- Apply recursive decent parser to generate parse tree.
5- Apply this program on this input.
   5 + 6 * 10

References:

1- Recursive decent parser http://web.cse.ohio-state.edu/software/2231/web-sw2/extras/slides/27.Recursive-Descent-Parsing.pdf
2- Recursive decent parser https://www.youtube.com/watch?v=S7DZdn33eFY
3- Left recursion https://www.youtube.com/watch?v=K-mfuhx9B_0
4- Left factoring https://en.wikipedia.org/wiki/LL_parser#Left_Factoring