Final lab parsing

TEAM:

CRISAN FILIP-DANIEL

CAMPEAN CATALIN-ALEXANDRU

Github link:

https://github.com/filipcrisan/ubb-lab-flcd/tree/main/lab-part-2

Documentation:

Algorithm: LL(1)

FIRST

- initialisation: for each non-terminal, add left-most element in production if it's a terminal or epsilon
- until F[i-1](A) = F[i](A) we keep iterating through the non-terminals, creating a set by concatenations of length 1

FOLLOW

- initialisation: for each non-terminal, add empty set and for the starting symbol add epsilon
- until F[i-1](X) = F[i](X) we keep iterating through the non-terminals, creating a set following multiple rules

PARSE TABLE:

- after FIRST and FOLLOW are generated, based on them we create the parse table
- we will have a table with columns being the terminals and rows being the non-terminals and terminals

- initialisation: everything set to err
- at every step we check if we are trying to set a cell again, which implies a conflict

Sequence parsing

- we do that using 2 stacks:
    - alpha → containing the sequence
    - beta → containing the natural flow of the grammar’s elements
- the goal is to reach the end of both of the stacks by matching the items in the stacks
- if the value in the parse table for the pair of the top elements is err, we found a syntax error
- the result will be a list of the indices of the right hand side productions

Tree generation

- based on the result of the sequence parsing, we create a tree
- every node will have a reference to its parent, its sibling and a certain value (either a non-terminal, or a terminal)