FLCD – Lab 6-7-8 – Documentation

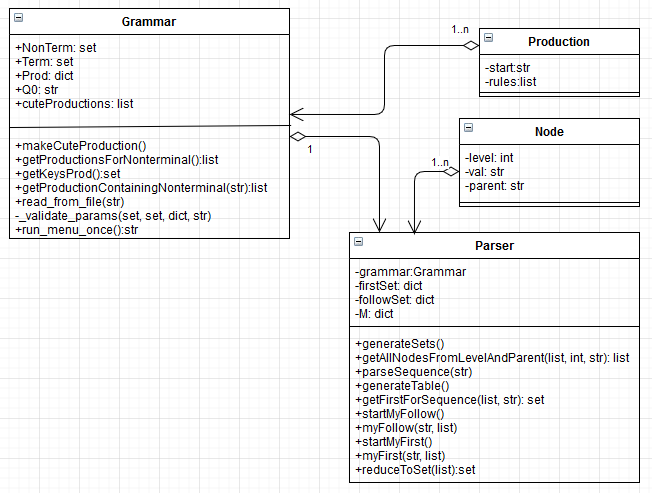
Lung Alin-Sebastian, Mărginean Lorena Simina

Gr. 934/2

Source code: <https://github.com/IcerOut/FLCD/tree/master>

Requirements:  
**Statement**: Implement a parser algorithm - LL(1)

Application Design:



Implementation details:

The Parser contains a grammar, based on which it builds two components, firstSet and followSet, which are used to create the parse table (M). firstSet starts being built in the function Parser.py > startMyFirst(), which calls myFirst() for each non-terminal, which calculates the first of that non-terminal recursively. followSet starts being built in the function Parser.py > startMyFollow(), which calls myFollow() for each non-terminal, which calculates the follow of that non-terminal recursively.

Afterwards, the parsing table is generated (initially filled by ERR) and gradually filled by the function generateTable(). If we reach a LL(1) conflict, we print an error message and stop the execution of the program.

parseSequence() can then be passed an input sequence (a string), converts it to a stack and then parses the sequence using the parsing table generated beforehand. If it reaches any invalid sequence (ERR in the table), it prints an error message, otherwise it prints a success message and the parsing tree.

Test cases:

Input: “g1.txt” – the program from the seminar

Sequences can be checked using the menu obtained by running “Main.py”

Input: “g2.txt” – our mini-language

Sequences can be checked using the menu obtained by running “Main.py”

Input: “g3.txt” – parser for addition, subtraction, multiplication and division with parantheses (from seminar, but extended)

Sequences can be checked using the menu obtained by running “Main.py”