Lab 5-6

Link Github: https://github.com/913-groza-vlad/Formal-Languages-and-Compiler-Design/tree/main/Lab5

The Grammar class represents a grammar and provides methods to initialize the grammar from a file, parse its content and check whether it is a context-free grammar.

The Grammar class has the following attributes:
- nonTerminals, which is a set containing the non-terminal symbols of the grammar
- terminals, which is a set containing the terminal symbols of the grammar
- productions, which is a map representing production rules, where each key is a non-terminal symbol, and the corresponding value is a set of production rules
- startSymbol, the start symbol of the grammar
- filename, the name of the file containing the grammar rules

The class contains methods for representing each of its fields in string format and the also the whole grammar (so we can display the grammar and its content) and the methods:

private Set<String> parseLine(String line):
This method parses a line from the grammar file to extract a set of elements enclosed in curly braces.

public boolean checkCFG():
Checks whether the grammar adheres to the rules of a context-free grammar. In order to return true, the following conditions must be met: the start symbol of the grammar must appear on the left-hand side of at least one production rule, all left-hand side symbols of production rules must be non-terminals and all symbols in the right-hand side must be either non-terminals, terminals, or the symbol epsilon.

public void readGrammarFromFile():
It reads the grammar rules from a specified file, initializes the grammar attributes (non-terminals, terminals, start symbol and productions) and populates the

Grammar object with the parsed information. The method reads the first two lines from the file, extracting non-terminals and terminals from lines enclosed in curly braces. Then, on the third line there is the start symbol of the grammar. Then, the productions are read by: skipping the line containing P = { header, then iterate through the remaining lines, parsing production rules and adding them to the set of productions for each non-terminal.

In the Parser class, there are implementations of the first and follow sets associated to each non-terminal of a grammar, which are essential for constructing the parsing table. Thus, these methods are described as:

public void computeFirst():
This method calculates the FIRST set for each non-terminal in the grammar and the algorithm iteratively refines the FIRST sets until no further changes occur. Initially, for each non-terminal A of the grammar, the set $F_0(A)$ is formed by checking those productions for the non-terminal A, whose right hand side starts with a terminal, and that terminal is added to $F_0(A)$. If there is a production for A that contains epsilon, it will also be added to the first set of A. Then, the method iterates over each non-terminal, updating its FIRST set based on the FIRST sets of its production symbols. For each production of the non-terminal, symbols are added to the FIRST set until a terminal or a symbol with a non-empty FIRST set is encountered. If the symbol has an epsilon in its FIRST set, it continues to the next symbol in the production. The iterations continue until $F_{i-1}(A) = F_i(A)$ for each non-terminal A and the last configuration of the FIRST is kept.

public void computeFollow():
It calculates the FOLLOW set for each non-terminal in the grammar. The FOLLOW set of a non-terminal is the set of terminals that can appear immediately to the right of instances of that non-terminal in some sentential form. The algorithm iteratively refines the FOLLOW sets until no further changes occur. We initialize for each non-terminal A of the grammar the set $L_0(A)$ with an empty set, excepting the follow set of the starting symbol, which contains epsilon. Then, for each non-terminal we identify all the productions which contains that terminal in the right hand side of the production and add in the $L_i(A)$ previous follow set: $L_{i-1}(A)$. If beta is the sequence that follows A in the right side of a production, the symbols in

FIRST(beta), excepting epsilon, are added to $L_i(A)$. If FIRST(beta) contains epsilon, or there is no sequence that follows A, in $L_i(A)$, the follow of the left hand side non-terminal is added. The iterations end when two columns are equal(the follow sets for all non-terminals doesn't change).