Formal Languages and Compiler Design

Lab 7 – Documentation

Ariadna Vilasó Escarp, Carlos Sánchez Rodríguez

Link to the repository:

https://github.com/UO270119/Formal-Languages-and-Compiler-Design

Problem Statement:

Implement a parser algorithm (final tests)

Input: 1) g1.txt + seq.txt

2) g2.txt + PIF.out (result of Lab 3)

Output: out1.txt, out2.txt

Run the program and generate:

- out1.txt (result of parsing if the input was g1.txt);

- out2.txt (result of parsing if the input was g2.txt)

Our task was to implement a parser algorithm using LL(1), which is a top-down parsing method. We chose to represent the solution as a table, using father and sibling relation.

To do this, we also had to develop the First and Follow functions, along with some classes.

We created five classes:

- *Grammar.java*, which is in charge of processing the input files and extract its set of terminals, non-terminals, productions and starting symbol. It also has its corresponding getters to access the properties of our grammar.
- *Pair.java,* which is an auxiliary class to create the map entries for the parsing table.
- *Production.java,* used to represent the production rules, with two private fields corresponding to the starting symbol and the list of rules, along with its corresponding getters.
- *Parser.java,* containing the implementation of the First and Follow algorithms and other methods such as *parseSequence()*, *createParseTable()*, and *initializeStacks()*, all of them used to create the parsing table.
- *ParserOutput.java,* used to represent the output of the Parser, whose most important methods are *put()* (inserts a new entry in the table), *get()* (retrieves an element from the table) and *containsKey()* (returns true if the given key is present in the table).

We also created a *Main.java* class that displays a menu with all the possible options in the application, which includes calculating the production for a given non-terminal, the first and follow sets, the parsing table and the parsing sequence for *g1.txt*. For our grammar in g1.txt, the sequence always has to start with an ‘a’. Knowing this, we can test our parser with the following sequences:

Accepting sequence: **a b b**

Non-accepting sequence: **b a b b**

This was developed by Ariadna Vilasó Escarp and Carlos Sánchez Rodríguez using pair programming.