PROJECT REPORT

1. Project Overview

This project consists of the implementation of two types of parsers for context-free grammars:

- LL(1) (Top-down parsing)

- SLR(1) (Bottom-up parsing)

The system accepts a user-defined grammar and input strings to determine if they belong to the language generated by the grammar. It was developed as part of the ST0270 Formal Languages and Compilers course.

2. Team Members

- Fabian Andres Buritica Cardozo

- Andres Felipe Toro Orrego

3. Environment and Tools

- Operating System: Windows 11

- Code Editor: Visual Studio Code

- Programming Language: Python 3.7+

4. How to Use the Program

1. Run the main Python file (`main.py`).

2. Enter the number of nonterminals in the grammar.

3. Input each production using the format: `NonTerminal -> production1 | production2 ...`

4. Use `e` (or `e`) to represent epsilon (empty string).

5. The system will identify if the grammar is LL(1), SLR(1), both or neither.

6. If applicable, the user selects a parser and tests input strings.

7. The program will return 'yes' if the string belongs to the language, and 'no' otherwise.

5. Implementation Details

The code implements algorithms to compute FIRST and FOLLOW sets.

It builds the LL(1) parsing table and SLR(1) parsing table, validating the grammar's properties before parsing.

6. Challenges and Solutions

During development, the main challenges included:

- Handling left recursion and common prefixes in LL(1)

- Building and handling SLR(1) states and conflict detection

Solutions involved clear grammar validation steps and structured parsing table construction.

7. Examples of Use

Example 1:

Grammar:

S -> S + T | T

T -> T * F | F

F -> (S) | i

Input: i+i

Output: yes

Example 2:

Grammar:

S -> AB

A -> aA | d

B -> bBc | e

Input: adbc

Output: yes

8. Compliance with Assignment Guidelines

This project follows the assignment guidelines:

- Developed by two students

- README.md provided

- Program meets input/output requirements

- Clear documentation and grammar verification steps implemented