Report

Name: Adil Aman Mohammed
Course: Formal Language Theory
Assignment No: 5
CWID: A20395630

Aim:
The main objective of this assignment is to design and implement a program capable of removing left recursion from a given Context-Free Grammar (CFG). The task involves the application of algorithms to identify and eliminate both immediate and indirect left recursion, resulting in a grammar that is suitable for top-down parsing.

Description:
The assignment entails writing a program that takes a CFG as an input, analyzes it to detect any left recursive rules, and transforms the grammar to an equivalent one without left recursion. The input CFG is supplied through a file, and the program must parse, process, and transform this grammar to meet the requirements.

Implementation:
Input and File Reading:
The program receives the path of an input file via command line arguments. The file, 'input.txt,' contains the CFG that needs left recursion removal. A second file, 'output.txt,' is utilized to store the transformed CFG.

Parsing the CFG:
The program reads the CFG from the input file and organizes its components into structured data types for ease of manipulation. The function _parse_grammar reads the file line by line, categorizing the non-terminals, terminals, and productions.

Left Recursion Removal:
Identifying Left Recursion:
The first step in the process is to identify any left recursive rules in the grammar. This is achieved through the function _find_left_recursive_rules, which scans through the productions to detect patterns of left recursion.

Removing Immediate Left Recursion:
Immediate left recursion is tackled using the function _remove_immediate_left_recursion. This function transforms any immediate left recursive rules into equivalent rules that do not exhibit left recursion.

Eliminating Indirect Left Recursion:
To handle indirect left recursion, the program employs the function _remove_indirect_left_recursion. This function reorders the non-terminals and substitutes productions as necessary to remove any indirect left recursion in the grammar.

Transformation and Verification:
The application of these algorithms transforms the CFG into a form that is devoid of left recursion, maintaining the integrity of the grammar and ensuring its compatibility with top-down parsing algorithms. The program verifies the correctness of the transformation, ensuring adherence to formal language theory principles.

Output:
The transformed CFG, now free of left recursion, is written to the ‘output.txt’ file. Each production is displayed clearly, showcasing the result of the left recursion removal algorithms. This output serves as a validation of the program’s effectiveness.

Conclusion:
This assignment was helpful in deepening the understanding of CFGs.