



# **University of Asia Pacific**

Department of Computer Science & Engineering

## **Compiler Design Lab**

**CSE 430**

### **Submitted to:**

Baivab Das

Lecturer

CSE, University of Asia Pacific

### **Submitted by:**

Asma Sultana

20101084

Section: B1

## Lab 3

### Problem

Elimination of Left Recursion in a grammar.

### Description of the problem

There is a production in the form of:

$$A \rightarrow A\alpha$$

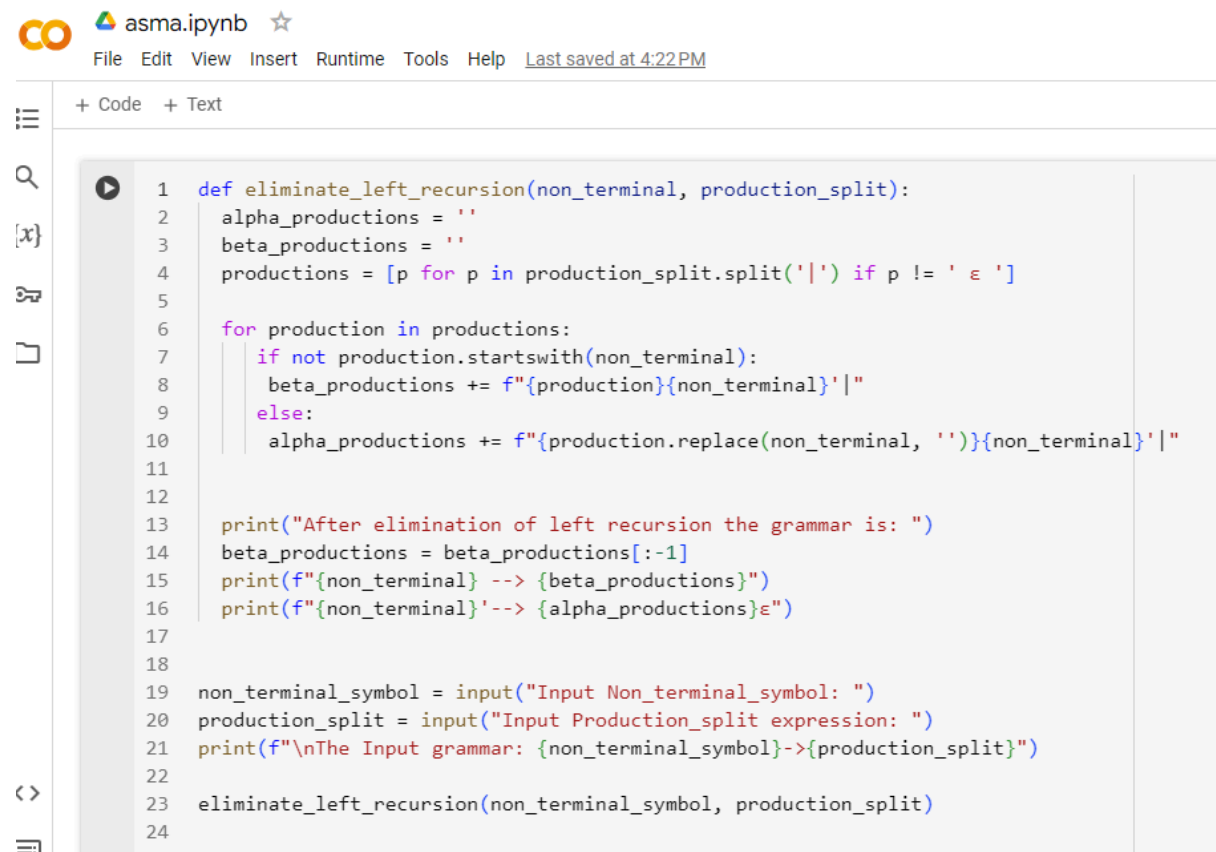
Now, we use the elimination left recursion algorithm to eliminate left recursion from a grammar which is given as an input in the console. Left recursion to consider

Now I generate the problem using this format-

$$A \rightarrow \beta A'$$

$$A' \rightarrow \alpha A' \mid \epsilon$$

### Code



The screenshot shows a Jupyter Notebook interface with a single code cell. The code defines a function `eliminate_left_recursion` that takes a non-terminal symbol and a production split expression as input. It processes the production split expression to generate new productions for the non-terminal, eliminating left recursion. The code includes comments and print statements to show the process and the resulting grammar.

```
1 def eliminate_left_recursion(non_terminal, production_split):
2     alpha_productions = ''
3     beta_productions = ''
4     productions = [p for p in production_split.split('|') if p != 'ε']
5
6     for production in productions:
7         if not production.startswith(non_terminal):
8             beta_productions += f"{production}{non_terminal}|"
9         else:
10            alpha_productions += f"{production.replace(non_terminal, '')}{non_terminal}|"
11
12
13    print("After elimination of left recursion the grammar is: ")
14    beta_productions = beta_productions[:-1]
15    print(f"{non_terminal} --> {beta_productions}")
16    print(f"{non_terminal}' --> {alpha_productions}ε")
17
18
19    non_terminal_symbol = input("Input Non_terminal_symbol: ")
20    production_split = input("Input Production_split expression: ")
21    print(f"\nThe Input grammar: {non_terminal_symbol}->{production_split}")
22
23    eliminate_left_recursion(non_terminal_symbol, production_split)
24
```

### Sample Input:

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

↶

Input Non\_terminal\_symbol: T

Input Production\_split expression:

### Observed Output

↶



Input Non\_terminal\_symbol: T

Input Production\_split expression: T \* F | F

The Input grammar:  $T \rightarrow T * F \mid F$

After elimination of left recursion the grammar is:

$T \rightarrow FT'$

$T' \rightarrow *F T' \mid \epsilon$

[ ] 1