Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Lab Report #5

Compiler Design

[Course Code: COMP 409]

[For the partial fulfillment of 4th year/1st Semester in Computer Engineering]

Submitted by:

Sabin Thapa

Roll no. 54

CE 4th Year

Submitted to:

Mr. Sushil Nepal

Assistant Professor

Department of Computer Science and Engineering

Submission date: Nov. 16, 2022

Task

Write a program to demonstrate left factoring from a left factored grammar.

Background

Left factoring is a process of factoring out the common prefixes of alternatives used when it is not clear that which of the alternatives is used to expand the non-terminal.

For a given grammar:

$$A \rightarrow \alpha \beta_1 \mid \alpha \beta_2$$

We can say that the grammar is left factored as the production alternatives have the same prefix α. So, inorder to remove the left factoring, we do the following:

$$A \rightarrow \alpha A'$$

$$A \rightarrow \beta_1 \mid \beta_2$$

Now, the grammar is free from left factoring.

Left factoring transforms the grammar such that it is useful for the Top-Down parsers. We came one production for each common prefixes and the rest of the derivation is added by new productions such that the parser has only one production per prefix which makes it easier for the parser to take decisions. Using this, back-tracing of the parser can be avoided.

Source Code

```
input_grammar = []
def take_grammar_input():
    input_grammar = []

    print('Note: \n 1. e represents epsilon. \n',
            '2. Represent in the form: A=>aB \n',
            '3. Press enter after each production. \n')

    while(True):
        user_input = input(f'Production: ')
        if user_input == '':
            break
        input_grammar.append(user_input)

    for i,production in enumerate(input_grammar):
        input_grammar[i]=production.split('=>')
        input_grammar[i][1]=input_grammar[i][1].split('|')
    i=0
    while i<len(input_grammar):
        j=0
        while j<len(input_grammar):
            if i!=j:
                if input_grammar[i][0] == input_grammar[j][0]:
                    input_grammar[i][1].extend(input_grammar[j][1])
                    input_grammar.pop(j)
            j+=1
        i+=1
    return input_grammar

def left_factoring():
    i=0
    while i<len(input_grammar):
        matched_strings = ''
        j = 0
        while j<len(input_grammar[i][1]):
            k=0
            while k<len(input_grammar[i][1]):
                if k!=j:
```

```
                    while True:
                        if len(input_grammar[i][1][k])>len(matched_strings):
                                                                                if
input_grammar[i][1][j][:len(matched_strings)+1]
 == input_grammar[i][1][k][:len(matched_strings)+1]:
                                                          matched_strings  +=
input_grammar[i][1][j][len(matched_strings)]
                                else:
                                    break
                            else:
                                break
                    k+=1
                j+=1
            if len(matched_strings):
                factors = []
                j=0
                while j<len(input_grammar[i][1]):
                    if len(matched_strings)<=len(input_grammar[i][1][j]):
                        if matched_strings == input_grammar[i][1][j]:
                            input_grammar[i][1].pop(j)
                            factors.append('e')
                                                      elif   matched_strings   ==
input_grammar[i][1][j][:len(matched_strings)]:
factors.append(input_grammar[i][1][j][len(matched_strings):])
                            input_grammar[i][1].pop(j)
                        else:
                            j+=1
                    else:
                        j+=1
                if (i+1)<len(input_grammar):
                    if input_grammar[i+1][0] == input_grammar[i][0]+"'":
input_grammar[i][1].insert(0,matched_strings+input_grammar[i][0]+"''")
input_grammar.insert(i+1,[input_grammar[i][0]+"''",factors])
                    else:
input_grammar[i][1].insert(0,matched_strings+input_grammar[i][0]+"'")
input_grammar.insert(i+1,[input_grammar[i][0]+"'",factors])
                else:
input_grammar[i][1].insert(0,matched_strings+input_grammar[i][0]+"'")
                    input_grammar.append([input_grammar[i][0]+"'",factors])
            i+=1
```

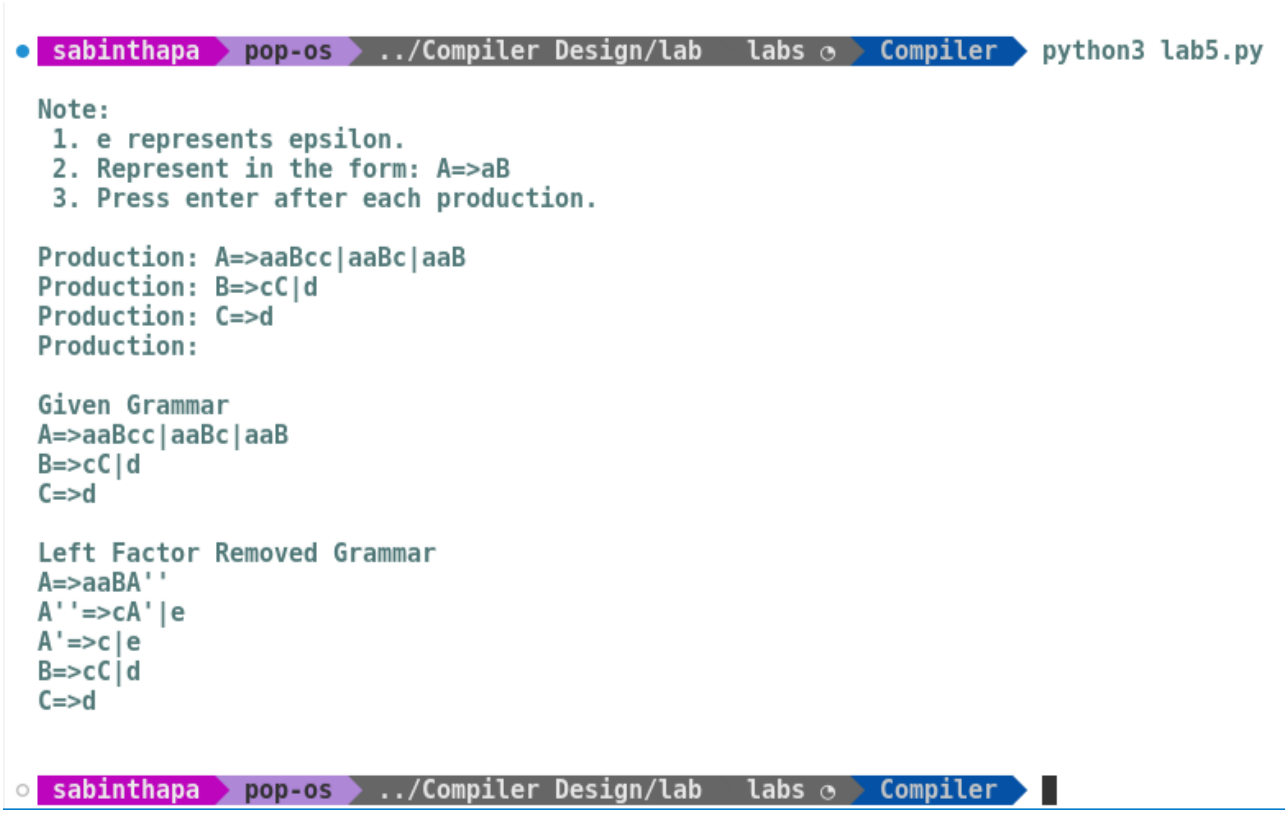
```
def print_output(grammar):
    for production in grammar:
        output_statement = ''
        output_statement+=production[0]+'=>'
        for j in range(len(production[1])):
            if j:
                output_statement+='|'
            output_statement+=production[1][j]
        print(output_statement)
    print()

if __name__=='__main__':
    input_grammar = take_grammar_input()
    print('\nGiven Grammar ')
    print_output(input_grammar)
    left_factoring()
    left_factoring()
    print('Left Factored Removed Grammar')
    print_output(input_grammar)
```

Explanation

When the user inputs a left factored grammar, the program checks for it and finds it and then it provides back the user with the grammar that has been freed from left factoring. There are three main functions:

1. take_grammar_input() to take the grammar input from the user.
2. left_factoring() to eliminate the left factoring from the grammar.
3. print_output() to display the output to the user.

Output



Here, the first production of the input grammar is left-factored. The aaB prefix of the alternatives is the same. After writing a separate production for the prefix, we can again see that, there is a presence of common prefix (A"c) again in the production. So, the production is again evaluated and the left factoring is removed. Finally, we get the productions that are free from left factoring.

Conclusion

In this way, the program to demonstrate and remove left factoring from the given left factored input grammar was implemented in Python. The source code and the outputs of the program are attached in the document above.