iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** IV **Month of publication:** April 2017

DOI: http://doi.org/10.22214/ijraset.2017.4175

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Implementation of Left Recursion using C Language

Faraah M. Dabhoiwala
Information Technology, Vadodara Institute of Engineering

Abstract: ***Compiler phases are divided into analysis and synthesis. Analysis phase comprises of lexical, syntax and semantic analysis. In this paper I have focused on removal of left recursion which is a pre requisite for most of the grammar that is to be input to the parser. I have implemented left recursion using C programming language.***
Keywords: Parser, left recursion, LL(k) parser, Context Free Grammar (CFG), C programme

I. INTRODUCTION

Parsing is the process of checking the syntax of a programming language (PL). If the syntax is correct then the parse tree is generated otherwise error is given by the compiler. It is one of the phases of compiler.
The syntax of any PL is checked using a Context Free Grammar (CFG). CFGs are most suitable for programming languages.
Parsing can either be top down or bottom up. In top down parsing approach the parser derives the string from the start symbol of the grammar.
Consider a CFG as shown below.
E->E+T|T
T->T*F|F
F->id
(Grammar 1)
Suppose we want to derive id + id * id.

E	->E+T	(using E->E+T)
	->T+T	(using E->T)
	->F+T	(using T->F)
	->id+T	(using F->id)
	->id+T*F	(using T->T*F)
	->id+F*F	(using T->F)
	->id+id*F	(using F->id)
	->id+id*id	(using F->id)

In the above case I have shown the successful derivation of the string. But this is not the case always.

II. LEFT RECURSION

Suppose we have grammar of the form
X->Xa1|Xa2|…|Xan|B1|B2|..|Bm (Grammar 2)
In this case the non terminal (NT) that appears on the left hand side is the first NT on the right hand side.
Due to this when parser chooses the production, it is possible that it might get into an infinite loop.
X->X a1
->Xa1 a1 (choosing X->Xa1)
->Xa1 a1 a1 (choosing X->Xa1)
->Xa1 a1 a1 a1 (choosing X->Xa1)
This substitution of X->Xa1 continues forever.

In order to overcome this situation we apply the following rules.

X->B1X'|B2X'|…|BmX'

X'->a1X'|a2X'|…|anX'| ϵ

Grammar 3

This removes left recursion.

From this we can say that Grammar 1 as shown above is also left recursive.

III.IMPLEMENTATION OF LEFT RECURSION USING C CODE

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
void main()
{
        char s[30],temp[20],LHS[30],fp[30],sp[30];
        int i,start,j,k=-1,flag1=0,flag2=0;
        clrscr();
        printf("enter production :");
        gets(s);
        start=2;
        LHS[0]=s[0];
        LHS[1]='\0';
        strcpy(fp,LHS);
        strcat(fp,"=");
        strcpy(sp,LHS);
        strcat(sp,"'=");
        r:
        j=0;
        for(i=start;s[i]!='|' && s[i]!='\0';i++)
        {
                temp[j]=s[i];
                j++;
        }
        temp[j]='\0';
        start=i+1;
        if(temp[0]==LHS[0])
        {
          if(flag1==1)
          {
                strcat(sp,"|");
          }
          strcat(sp,a+1);
          strcat(sp,LHS);
          strcat(sp,"'");
          flag1=1;
        }
        else
        {
          if(flag2==1)
          {
```

```
                strcat(fp,"|");
              }
              strcat(fp,a);
              strcat(fp,LHS);
              strcat(fp,"'");
              flag2=1;
            }
      if(start<strlen(s))
      {
            goto r;
      }
      strcat(sp,"|null");
      printf("\n%s",fp);
      printf("\n%s",sp);
      getch();
}
```

IV. ALGORITHM

Step 1: Take the production of Context Free Grammar (CFG) from the user for which left recursion has to be removed.
Step 2: As CFGs have only one NT on LHS we can start reading the production from the third symbol. Let start be a variable and start=2 as array index starts from 0.
Step 3: Take a string variable named LHS and store the NT on LHS of the given production in that string.
Step 4: Copy the NT obtained in the previous step in two strings, fp and sp and append '=' symbol to both the strings.
Step 5: Read the production from the position marked by start variable till '|' symbol is found or End of String is encountered and copy character by character in a temporary string variable i.e. temp
Step 6: Let position of start variable is the symbol after '|' sign in the given production.
Step 7: Check whether the first character of production obtained in the string temp is same as the character in the string variable LHS. If yes then go to step 8 else go to step 9.
Step 8: If it is the first option in production stored in string sp then don't append '|'sign. If it is other than first option then append '|' symbol. Also append the string after the first character in the string temp. Then append string in LHS variable with a single quote (') sign added to it and go to step 10.
Step 9: If it is the first option in production stored in string fp then don't append '|'sign. If it is other than first option then append '|' symbol. Also append the string after the first character in the string temp. Then append string in LHS variable with a single quote (') sign added to it and go to step 10.
Step 10: If value in start variable is less than the length of the string given by the user then go to step 5 else go to step 11.
Step 11: Append the string "|ϵ" to sp.
Step 12: Print fp and sp.

V. CONCLUSION

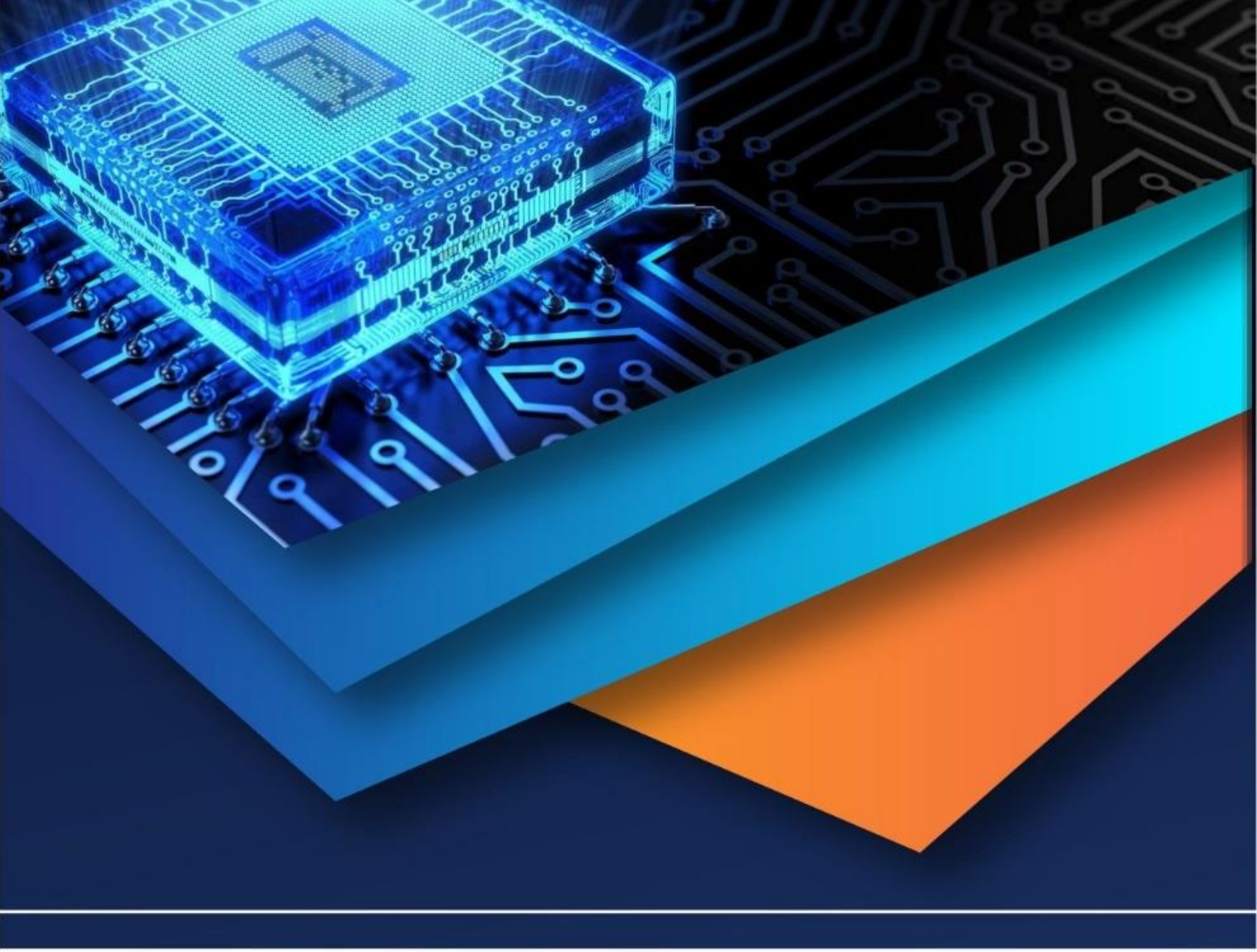
Hence in the above paper I have implemented removal of left recursion using C programming language.

VI. ACKNOWLEDGMENT

I would like to thank my Parents for their endless support and Mr. Rahil Barafwala, for constantly encouraging me to work on research paper and this work is the result of the same.

REFERENCES

[1] Dhamdhere D M, Systems Programming and Operating Systems.
[2] Aho A. V., Sethi R., Ulman J.D., Compilers Principles, Techniques and Tools.
[3] FARAAH M. DABHOIWALA, "A REVIEW ON LL(1) PARSER", INTERNATIONAL JOURNAL FOR RESEARCH IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY (IJRASET), VOLUME 2 ISSUE XI, NOVEMBER 2014.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY