



常州大学
CHANGZHOU UNIVERSITY

实 验 报 告

第 二 次实验

课程名称: 编译原理

实验名称: 文法的推导

学 院: 信息数理学院 班 级: 计算机 162

学 号: 16430121 姓 名: 林锦雄

完成日期: 2019.5.30

指导教师: 顾玉宛 成 绩:

一、实验目的

输入文法和符号串，按照所提供的要求，是左推导还是右推导，输出由该文法推导该符号串的推导结果。输出从识别符到当前句型的所有推导的具体推导过程，如果推导成功，该符号串是文法的句子。

二、实验过程

1.试验准备：在学习了文法和推导的一些基本概念后，用本实验来加深对左推导和右推导的认识，和符号串是文法的句子的理解。

2.数据结构：使用结构体来定义数据的结构，用链表的形式来保存文法以及文法之间的关系。

3.代码编写：

```
1. #include<iostream.h>
2. #include<String.h>
3.
4. struct LeftItem;
5. struct RightNode { //存储规则右部的链表结表结构
6.     char right;
7.     char sign;
8.     RightNode* nextsibling;
9.     RightNode* nextrule;
10.     RightNode(char abc,int s=1) {
11.         right=abc;
12.         nextsibling=NULL;
13.         nextrule=NULL;
14.         sign=s;
15.     }
16. };
17.
18. struct LeftItem { //存储规则左部的数组元素结构
19.     char left;
20.     RightNode* therule;
21. };
22.
23. void Insert(RightNode*& pNode,char* temp) { //将规则右部字符插入链表中
24.     pNode=new RightNode(*temp);
25.     RightNode* qNode=pNode;
26.     temp++;
27.     while(*temp!='\0') {
28.         qNode->nextsibling=new RightNode(*temp);
29.         qNode=qNode->nextsibling;
30.         temp++;
31.     }
32. }
33.
34. void Initial(LeftItem Array[],int size,RightNode* pNode) { //对规则右部节点的标志
    置初值,指明是终结或非终结符
35.     RightNode* qNode=pNode->nextrule;
36.     while(pNode!=NULL){
```

```
37.         for(int i=0; i<size; i++){
38.             if(pNode->right==Array[i].left)
39.                 break;
40.         }
41.         if(i==size){
42.             pNode->sign=0;
43.         }
44.         pNode=pNode->nextsibling;
45.     }
46.     if(qNode!=NULL)
47.         Initial(Array,size,qNode);
48. }
49.
50. int Select1(LeftItem Array[],RightNode* pNode,int size){//左推导时,求句型中下一个
被替换的非终结符
51.     while(pNode!=NULL){
52.         if(pNode->sign==1)
53.             break;
54.         pNode=pNode->nextsibling;
55.     }
56.     if(pNode!=NULL){
57.         for(int i=0; i<size; i++){
58.             if(Array[i].left==pNode->right)
59.                 return i;
60.         }
61.     }
62.     else{
63.         return -1;
64.     }
65. }
66.
67. int Select2(LeftItem Array[],RightNode* pNode,int size){ //右推导时,求句型中下一个
被替换的非终结符
68.     while(pNode!=NULL){
69.         if(pNode->sign==1)
70.             break;
71.         pNode=pNode->nextrule;
72.     }
73.     if(pNode!=NULL){
74.         for(int i=0; i<size; i++){
75.             if(Array[i].left==pNode->right)
76.                 return i;
77.         }
78.     } else{
79.         return -1;
80.     }
81. }
82.
83. void Bianli(RightNode* pNode,char F,int& index,RightNode* NArray[]){//输出当前要
被替的非终结符对应的所有规则
84.     NArray[index]=pNode;
85.     cout<<index+1<<". "<<F<<":=";
86.     RightNode* qNode=pNode->nextrule;
87.     while(pNode!=NULL){
88.         cout<<pNode->right;
89.         pNode=pNode->nextsibling;
90.     }
91.     if(qNode!=NULL){
```

```
92.         cout<<endl;
93.         index++;
94.         Bianli(qNode,F,index,NArray);
95.     }
96. }
97.
98. void Change1(RightNode* pNode,RightNode*& first,RightNode*& last){//左推导时替换
句型中当前非终结符
99.     RightNode* p=first;
100.    RightNode* q=first;
101.    RightNode* r=pNode;
102.    if(first->sign==1){
103.        p=first=new RightNode(r->right,r->sign);
104.        r=r->nextsibling;
105.    } else {
106.        while(p!=NULL){
107.            if(p->nextsibling->sign==1)
108.                break;
109.            p=p->nextsibling;
110.        }
111.        q=p->nextsibling;
112.    }
113.    while(r!=NULL){
114.        p->nextsibling=new RightNode(r->right,r->sign);
115.        r=r->nextsibling;
116.        p=p->nextsibling;
117.    }
118.    p->nextsibling=q->nextsibling;
119.    if(q->nextsibling==NULL)
120.        last=p;
121. }
122.
123. void Change2(RightNode* pNode,RightNode*& first,RightNode*& last){//右推导时,替
换句型中当前非终结符
124.    RightNode* p=first;
125.    RightNode* q=last;
126.    RightNode* r=pNode;
127.    while(q!=NULL){
128.        if(q->sign==1)
129.            break;
130.        q=q->nextrule;
131.    }
132.    if(q==first){
133.        p=first=new RightNode(r->right,r->sign);
134.        r=r->nextsibling;
135.    } else
136.        p=q->nextrule;
137.    while(r!=NULL){
138.        p->nextsibling=new RightNode(r->right,r->sign);
139.        p->nextsibling->nextrule=p;
140.        r=r->nextsibling;
141.        p=p->nextsibling;
142.    }
143.    p->nextsibling=q->nextsibling;
144.    if(q->nextsibling==NULL)
145.        last=p;
146.    else
147.        q->nextsibling->nextrule=p;
```

```
148. }
149.
150. void Display(RightNode* pNode, char TD[]) { //输出从识别符到当前句型的所有推导
151.     char temp[30];
152.     temp[0] = '=';
153.     temp[1] = '>';
154.     int i = 2;
155.     while (pNode != NULL) {
156.         temp[i] = pNode->right;
157.         i++;
158.         pNode = pNode->nextsibling;
159.     }
160.     temp[i] = '\n';
161.     temp[i+1] = ' ';
162.     temp[i+2] = '\0';
163.     strcat(TD, temp);
164.     cout << TD;
165. }
166.
167. void main() {
168.     char temp[30]; //建立字符数组暂存用户输入的文法规则和要
//推导句型
169.     char TD[500]; //建立字符数组存放从识别符到当前句型的所
有推导
170.     char p[20];
171.     char temp1[30];
172.     int size = 0;
173.     int i, j, k, n;
174.     int sel1, sel2;
175.     char sel3;
176.     int index1, index2;
177.     RightNode* NArray[10];
178.     RightNode* first; //建立链表存放推导过程中的句型
179.     RightNode* last;
180.     RightNode* q;
181.     cout << "请输入文法规则的数目: ";
182.     cin >> n;
183.     LeftItem* Array = new LeftItem[n];
184.     for (i = 0; i < n; i++) {
185.         cout << "请输入文法规则: ";
186.         cin >> temp;
187.         for (j = 0; j < size; j++) { //判别要输入规则的左部有没有存储过
188.             if (Array[j].left == *temp) {
189.                 k = j;
190.                 break;
191.             }
192.         }
193.         if (j == size) { //要输入规则的左部没有存储过
194.             Array[size].left = *temp;
195.             Insert(Array[size].therule, temp + 4);
196.             size++;
197.         } else { //要输入规则的左部已存储
198.             RightNode* t = Array[k].therule;
199.             while (t->nextrule != NULL)
200.                 t = t->nextrule;
201.             Insert(t->nextrule, temp + 4);
202.         }
203.     }
```

```
204.     for(i=0; i<size; i++)
205.         Initial(Array,size,Array[i].therule);
206.     cout<<endl<<"文法规则存储完毕!"<<endl<<endl;
207. aaa:
208.     TD[0]=Array[0].left;
209.     TD[1]='\0';
210.     cout<<"请输入要推导的符号串:";
211.     cin>>temp;
212.     cout<<"----左推导请按----1"<<endl;
213.     cout<<"----右推导请按----2"<<endl;
214.     cout<<"----退出请按-----3"<<endl;
215.     do{
216.         cin>>sel1;
217.     }
218.     while(sel1!=1&&sel1!=2&&sel1!=3);
219.     if(sel1!=3){
220.         first=last=new RightNode(Array[0].left);
221.         cout<<Array[0].left<<"=">";
222.         while(1){
223.             cout<<endl;
224.             index2=0;
225.             if(sel1==1)
226.                 index1=Select1(Array,first,size); //左推导时取得当前要替换的非终结
符在左部数组中的编号
227.             else
228.                 index1=Select2(Array,last,size); //右推导时取得当前要替换的非终结符
在左部数组中的编号
229.             if(index1==-1)
230.                 break;
231.             Bianli(Array[index1].therule,Array[index1].left,index2,NArray); //显
示要替换的规则
232.             cout<<endl<<endl;
233.             cout<<"要推导的符号串为: ";
234.             cout<<temp;
235.             cout<<endl;
236.             cout<<"请选择要替换的规则编号(放弃推导请按0):";
237.             do{
238.                 cin>>sel2;
239.             }
240.             while(sel2<0||sel2>index2+1);
241.             if(sel2==0)
242.                 break;
243.             if(sel1==1)
244.                 Change1(NArray[sel2-1],first,last); //左推导时替换当前的非终结符
245.             else
246.                 Change2(NArray[sel2-1],first,last); //右推导时替换当前的非终结符
247.             Display(first,TD);
248.         }
249.         i=0;
250.         q=first;
251.         while(q!=NULL){
252.             temp1[i]=q->right;
253.             q=q->nextsibling;
254.             i++;
255.         }
256.         temp1[i]='\0';
257.         if(strcmp(temp,temp1)==0){
258.             cout<<"推导成功!"<<endl;
```

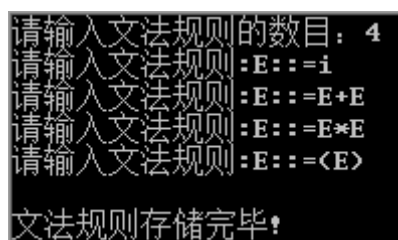
```

259.         cout<<"该符号串是文法的句子!"<<endl<<endl;
260.         cout<<"具体推导过程如下:"<<endl;
261.         cout<<TD;
262.     }
263.     else
264.         cout<<"推导失败!"<<endl<<endl;
265.     cout<<"想再试一次吗? ";
266.     do{
267.         cin>>sel3;
268.     }
269.     while(sel3!='Y'&&sel3!='y'&&sel3!='N'&&sel3!='n');
270.     if(sel3=='Y' || sel3=='y'){
271.         cout<<endl<<endl;
272.         goto aaa;
273.     }
274. }
275. }

```

三、实验结果与分析

1. 文法规则存储:



```

请输入文法规则的数目: 4
请输入文法规则: E::=i
请输入文法规则: E::=E+E
请输入文法规则: E::=E*E
请输入文法规则: E::=(E)
文法规则存储完毕!

```

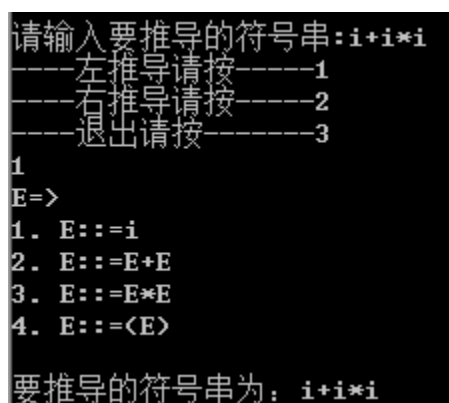
图2 文法规则存储

输入包含四条规则的一组文法规则:

- E->i
- E->E+E
- E->E*E
- E->(E)

成功录入文法后，系统会有“文法规则存储完毕!”的提示语。如图2所示。

2. 左推导:



```

请输入要推导的符号串: i+i*i
----左推导请按-----1
----右推导请按-----2
----退出请按-----3
1
E=>
1. E::=i
2. E::=E+E
3. E::=E*E
4. E::=(E)
要推导的符号串为: i+i*i

```

图3 左推导

输入要推导的符号串： $i+i*i$ 。然后在子菜单中选择左推导。如图 3 所示。

```

请选择要替换的规则编号<放弃推导请按0>:2
E=>E+E

1. E::=i
2. E::=E+E
3. E::=E*E
4. E::=(E)

要推导的符号串为: i+i*i
请选择要替换的规则编号<放弃推导请按0>:1
E=>E+E
=>i+E

1. E::=i
2. E::=E+E
3. E::=E*E
4. E::=(E)

要推导的符号串为: i+i*i
请选择要替换的规则编号<放弃推导请按0>:3
E=>E+E
=>i+E
=>i+E*E

1. E::=i
2. E::=E+E
3. E::=E*E
4. E::=(E)

要推导的符号串为: i+i*i
请选择要替换的规则编号<放弃推导请按0>:1
E=>E+E
=>i+E
=>i+E*E
=>i+i*E

1. E::=i
2. E::=E+E
3. E::=E*E
4. E::=(E)

要推导的符号串为: i+i*i
请选择要替换的规则编号<放弃推导请按0>:1
E=>E+E
=>i+E
=>i+E*E
=>i+i*E
=>i+i*i

推导成功!
该符号串是文法的句子!

具体推导过程如下:
E=>E+E
=>i+E
=>i+E*E
=>i+i*E
=>i+i*i
想再试一次吗? y

```

图 4 左推导过程

左推导过程：最初为 E；

- 1) 由 $E \Rightarrow E+E$ ，把最初的 E 进行推导，推导出 $E \Rightarrow E+E$;
- 2) 由 $E \Rightarrow i$ ，把右边中最左边的 E 进行置换，得到 $E \Rightarrow i+E$;
- 3) 由 $E \Rightarrow E * E$ ，把右边最后一个 E 进行置换，推导出 $E \Rightarrow i+E * E$;
- 4) 由 $E \Rightarrow i$ ，把右边中最左边的 E 进行置换，得到 $E \Rightarrow i+i * E$;
- 5) 由 $E \Rightarrow i$ ，把右边中最左边的 E 进行置换，得到 $E \Rightarrow i+i * i$;

左推导成功，该符号串是文法的句子。系统会自动把以上左推导过程整理好再列出来。最后询问是否继续对符号串进行推导。具体的左推导过程，如图 4 所示。

3.右推导:

```
想再试一次吗? y
请输入要推导的符号串: i+i*i
----左推导请按-----1
----右推导请按-----2
----退出请按-----3
2
E=>
1. E::=i
2. E::=E+E
3. E::=E * E
4. E::=<E>
要推导的符号串为: i+i*i
```

图 5 右推导

输入要推导的符号串: $i+i*i$ 。然后在子菜单中选择右推导。如图 5 所示。

```

请选择要替换的规则编号<放弃推导请按0>:2
E=>E+E

1. E::=i
2. E::=E+E
3. E::=E*E
4. E::=(E)

要推导的符号串为: i+i*i
请选择要替换的规则编号<放弃推导请按0>:3
E=>E+E
=>E+E*E

1. E::=i
2. E::=E+E
3. E::=E*E
4. E::=(E)

要推导的符号串为: i+i*i
请选择要替换的规则编号<放弃推导请按0>:1
E=>E+E
=>E+E*E
=>E+E*i

1. E::=i
2. E::=E+E
3. E::=E*E
4. E::=(E)

要推导的符号串为: i+i*i
请选择要替换的规则编号<放弃推导请按0>:1
E=>E+E
=>E+E*E
=>E+E*i
=>E+i*i

1. E::=i
2. E::=E+E
3. E::=E*E
4. E::=(E)

要推导的符号串为: i+i*i
请选择要替换的规则编号<放弃推导请按0>:1
E=>E+E
=>E+E*E
=>E+E*i
=>E+i*i
=>i+i*i

推导成功!
该符号串是文法的句子!

具体推导过程如下:
E=>E+E
=>E+E*E
=>E+E*i
=>E+i*i
=>i+i*i
想再试一次吗?

```

图 6 右推导过程

右推导过程：最初为 E；

6) 由 $E \Rightarrow E+E$ ，把最初的 E 进行推导，推导出 $E \Rightarrow E+E$ ；

7) 由 $E \Rightarrow E*E$ ，把右边最后一个 E 进行置换，推导出 $E \Rightarrow E+E*E$ ；

8) 由 $E \Rightarrow i$ ，把右边中最左边的 E 进行置换，得到 $E \Rightarrow E+E*i$ ；

9) 由 $E \Rightarrow i$, 把右边中最左边的 E 进行置欢, 得到 $E \Rightarrow E+i*i$;

10) 由 $E \Rightarrow i$, 把右边中最左边的 E 进行置欢, 得到 $E \Rightarrow i+i*i$;

右推导成功, 该符号串是文法的句子。系统会自动把以上右推导过程整理好再列出来。最后询问是否继续对符号串进行推导。具体的右推导过程, 如图 6 所示。

四、实验心得

通过本次“文法的推导”实验, 加深了我对文法的各种推导方法的理解, 尤其是文法的左推导和右推导方法的具体过程。

输入文法和符号串, 选择是左推导还是右推导, 一步一步对符号串按照相应的规则进行推导, 最后输出从识别符到当前句型的所有推导的具体推导过程, 如果推导成功, 该符号串是文法的句子。

在学习了规则和有关文法的推导后, 用本实验来加深各个概念间的关系。使用结构体来定义数据的结构, 用一种类似链表的形式来保存文法以及文法之间的关系。