



合肥工业大学
HEFEI UNIVERSITY OF TECHNOLOGY

编译原理实验报告

学生姓名：林天岳

学 号：2017217893

班 级：计算机科学与技术 2017-5

完成日期：2019 年 10 月 22 日

实验 2：LL(1)分析法

1. 数据结构及算法描述

```
2. Set<String> noTerminal = new HashSet<>();//非终结符
3. Set<String> terminal = new HashSet<>();//终结符
4. Map<String,Set<String>> First = new HashMap<>();//First 集
5. Map<String,Set<String>> Follow = new HashMap<>();//Follow 集
6. Map<String,Set<String>> select = new HashMap<>();//产生式的 select 集
7. List<String[]> Symbol_Gram = new ArrayList<>();// { { 符号 , 存在的文法 } *n }
8. //使用 set 保存数据,确保无重复元素 在计算的时候无序手动排除重复元素
```

```
1. 传入文法 G
2.
3. G 按照\n 和"->"以及"\\"分割为单元
4.
5. if(成功){
6.     保存并更新语法
7. }
8. else{
9.     弹出语法错误警告
10. }
11. if(存在左递归){
12.     弹出左递归警告
13. }
14.
15. 计算 First 集(){
16.     for(String symbol:终结符){
17.         First(symbol) = [symbol]
18.     }//终结符的 First 集是本身
19.     while(First 集的大小还在变化){
20.         for(String 左边->右边 :所有文法){
21.             取出右边下标为 0 的符号
22.             if(当前符号为非终结符){//当前符号的 First 集除了空都加入到左边符号的 First
                集中
23.                 First(左边).addAll(First(当前符号).except("ε"));//
24.                 if(当前符号的 First 集合含有空){
25.                     看向下一个符号 //即下标+1 递归处理
26.                 }
```

```

27.         }
28.         else if(当前符号是终结符 或者 "ε"){
29.             把当前符号加入到左边符号的 First 集中
30.         }
31.         else if(是"\0"){
32.             停止
33.         }
34.         else{
35.             停止
36.         }
37.     }
38. }
39. }
40.
41. 计算 Follow 集(){
42.     在开始符号的 Follow 集中加入 "#"
43.     for(String 当前符号:非终结符){
44.         for(语法 当前语法:所有的含有当前计算 Follow 集符号的语法){
45.             String 紧跟符号 = 当前语法中,当前符号之后的一个符号
46.             if(紧跟符号为终结符){
47.                 把紧跟符号加到当前符号的 Follow 集中
48.             }
49.             else if(紧跟符号为非终结符){
50.                 把紧跟符号的 First 集-"ε"加入到当前符号的 Follow 集中
51.                 if(当前符号可以的 First 集含有空)
52.                     看向下一个符号//也是递归求解
53.             }
54.             else if(当前符号是"\0"){
55.                 把空加入到当前符号的 Follow 集
56.             }
57.             else{
58.                 报错 停止
59.             }
60.         }
61.     }
62. }
63.
64. 计算 Select 集(){
65.     for(String 当前文法(左边->右边) :所有文法){
66.         String 当前符号 = 右边的第一个符号
67.         if(当前符号是终结符){
68.             把当前符号加入 Select(当前文法)
69.         }
70.         else if(当前符号是"ε"或者是"\0"){

```

```

71.         把 Follow(左边)加入到 Select(当前文法)
72.     }
73.     else if(当前符号是非终结符){
74.         把 First(当前符号).except("ε")加入到 Select(当前文法)中
75.         if(当前符号的 First 集含有"ε")
76.             看向下一个符号 递归求解
77.     }
78.     else{
79.         报错 停止
80.     }
81. }
82. }
83.
84. 计算 M 表(){
85.     for(String 当前文法:select 集){
86.         for(String 当前符号:Select(当前文法)){
87.             M(当前文法 的 左边,当前符号) = 当前文法
88.         }
89.     }
90. }
91.
92. 分析过程(String 输入的内容){
93.     初始化分析栈
94.     初始化输入栈
95.     while(结束标记为未结束){
96.         if(存在文法){
97.             if(M(x,a) == "ε"){
98.                 分析栈出栈
99.             }
100.        else{
101.            分析栈.push(M(分析栈.pop(),a))
102.        }
103.    }
104.    else if(匹配到了 且没有结束){
105.        输入栈.pop()
106.        分析栈.pop()
107.    }
108.    else if(匹配成功 是#){
109.        结束标记修改为结束
110.    }
111.    else{
112.        报错
113.        结束标记修改为结束
114.    }

```

```

115.      }
116.  }

```

2.算法流程图



LL(1)预测分析程序流程

3. 源码及测试结果

Main.java :

```
1. package 实验二__LL1 分析法;
2.
3. public class Main {
4.     public static void main(String[] args) {
5.         Windows windows = new Windows();
6.     }
7. }
```

Solution.java

```
1. package 实验二__LL1 分析法;
2. import java.util.*;
3. import java.util.List;
4. import java.util.stream.Collectors;
5.
6. class Solution {
7.     Map<String,String> AnaTable = new HashMap<>();
8.     Map<String,Set<String>> select = new HashMap<>();//产生式的 select 集
9.     List<String[]> Symbol_Gram = new ArrayList<>();// { { 符号 , 存在的文
    法 } *n }
10.     String x;
11.     String a;
12.     Set<String> noTerminal = new HashSet<>();
13.     Set<String> terminal = new HashSet<>();
14.     Set<String> allG = new HashSet<>();
15.     Map<String,Set<String>> First = new HashMap<>();
16.     Map<String,Set<String>> Follow = new HashMap<>();
17.     String GStart = "null";
18.     Solution(){
19.         setG("E -> TG \n" +
20.             "G -> +TG | -TG \n" +
21.             "G -> ε \n" +
22.             "T -> FS \n" +
23.             "S -> *FS | /FS \n" +
24.             "S -> ε \n" +
25.             "F -> (E) \n" +
26.             "F->i \n");
```

```

27.     }
28.     String setG(String G){
29.         Map<String,String> AnaTable = new HashMap<>();
30.         Symbol_Gram.clear();
31.         G = G.replaceAll(" ", "");
32.         String []Gs = G.split("\n");
33.         if(Gs.length<1){
34.             return "输入错误";
35.         }
36.         GStart = Gs[0].substring(0,1) ;
37.         for (String gLine : Gs) {
38.             if( gLine.split("->").length!=2){
39.                 return "格式错误";
40.             }
41.             String split0 = gLine.split("->")[0];
42.             for (String str : gLine.split("->")[1].split("\\|")) {
43.                 String [] SingleG = {split0,str};
44.                 Symbol_Gram.add(SingleG);
45.             }
46.         }
47.         GStart = Gs[0].substring(0,1) ;
48.         MyStack stack = new MyStack();
49.         for (String[] strings : Symbol_Gram) {
50.             if(strings[0].equals(strings[1].substring(0,1))){
51.                 return "左递归";
52.             }
53.         }
54.         getFF();//计算 First Follow 集
55.         for (String[] strings : Symbol_Gram) {
56.             select.put(strings[0]+strings[1],new HashSet<>());
57.         }
58.         //例如 s->ab strings[0] -> strings[1]
59.         for (String[] strings : Symbol_Gram) {
60.             setSelect(strings,0); //创建 select 集 key = 产生
           式 value = [] Set
61.         }
62.         AnaTable.clear();//清空
63.         terminal.add("#");//在终结符内加入# 以达到
64.         for (String s0 : select.keySet()) {
65.             for (String s1 : select.get(s0)) {
66.                 AnaTable.put(s0.substring(0,1)+s1,s0.substring(1));
67.             }
68.         }
69.         this.AnaTable =AnaTable;

```

```

70.         return null;
71.     }
72.     Vector<String[]> Solve(String text){
73.         Vector<String[]> procesList = new Vector();
74.         int textLength = text.length();
75.         MyStack AnaStack = new MyStack();//分析栈
76.         MyStack inputString = new MyStack();//输入串
77.         AnaStack.push("#","E","S");
78.         inputString.push( new StringBuffer(text).reverse().toString().split(
            "" ) );
79.         Boolean flag = true;
80.         Boolean matched = true;
81.         int linenumber = 0;
82.         while (flag){
83.             String[] INFO = new String[5];
84.             INFO[0] = String.valueOf(linenumber++);
85.             INFO[1] = AnaStack.toString();
86.             StringBuffer inputsb= new StringBuffer(inputString.toString());
87.             for(int sblength = inputsb.length() ; sblength < textLength+1 ;
                sblength++ ){
88.                 inputsb.append(" ");
89.             }
90.             INFO[2] = inputsb.reverse().toString();
91.             x = AnaStack.getTop();//获取分析栈顶
92.             a = inputString.getTop();//第一个符号读到 a
93.             if(M(x,a)!=null){//存在对应的文法
94.                 if(M(x,a)[0].equals("ε")){//为空
95.                     INFO[3] = x+" -> ε";
96.                     INFO[4] = "POP";
97.                     AnaStack.pop();
98.                 }
99.                 else {
100.                     AnaStack.push(M(AnaStack.pop(), a));
101.                     StringBuffer sb = new StringBuffer();
102.                     for(String string:M(x,a)){
103.                         sb.append(string);
104.                     }
105.                     INFO[3] = x+" -> " + sb.reverse();
106.                     INFO[4] = "POP,PUSH("+sb.reverse()+")";
107.                     //System.out.println("存在文法
                        ["+x+", "+a+"] -> ["+sb + " ] STACK:" + AnaStack);
108.                 }
109.             }

```



```

110.         else if( !x.equals("#") && x.equals(a) ){//匹配到了
111.             //System.out.println("匹配到了"+x+" "+a);
112.             INFO[4] = "POP,GETNEXT(i)";
113.             inputString.pop();
114.             AnaStack.pop();
115.         }
116.         else if(x.equals("#") && x.equals(a)){//结束了
117.             flag=false;
118.         }
119.         else{//报错
120.             flag=false;
121.             matched=false;
122.         }
123.         for(int i = 0 ; i < 5 ; i++ ){
124.             if(INFO[i]!=null)
125.                 INFO[i] = " "+INFO[i];
126.             else
127.                 INFO[i] = " ";
128.         }
129.
130.         procesList.add(INFO);
131.     }
132.     if(matched){
133.         //System.out.println("匹配成功");
134.     }
135.     else{
136.         //System.out.println("匹配失败");
137.         String ss[] ={"ERROR","ERROR","ERROR","ERROR","ERROR"};
138.         procesList.add(ss);
139.     }
140.
141. //     for (String s : Grammer) {
142. //         System.out.print(s+" ");
143. //     }
144.     return procesList;
145. }
146. String[] M(String Line ,String column){
147.     //倒序 分割
148.     //System.out.println("查询 " + Line + "<->" + column);
149.     if(AnaTable.get(Line+column) == null) {
150.         if ( AnaTable.get(Line+"#")!=null) {
151. //             System.out.println("返回空");
152.             String ss[] = {"ε"};
153.             return ss;

```

```

154.         }
155.         else {
156.             return null;
157.         }
158.     }
159.     else{
160.         return new StringBuffer(AnaTable.get(Line+column)).reverse().to
String().split("");
161.     }
162. }
163. Map[] getFF(){
164.     noTerminal.clear();
165.     First.clear();
166.     Follow.clear();
167.     allG.clear();
168.     terminal.clear();
169.     for (String[] strings : Symbol_Gram) {
170.         noTerminal.add(strings[0]);
171.     }//非终结符
172.     for (String[] strings : Symbol_Gram) {
173.         allG.add(strings[0]);
174.         for (String s : strings[1].split("")) {
175.             allG.add(s);
176.         }
177.     }//所有符
178.     allG.remove("ε");
179.     terminal.addAll(allG.stream().filter(S-> !noTerminal.contains(S)).
collect(Collectors.toSet()));//终结符 = 所有符号 - 非终结符
180.     //System.out.println(Grammer+"\n"+EndG);
181.     for (String s1 : allG) {
182.         First.put(s1,new HashSet<>());
183.         Follow.put(s1,new HashSet<>());
184.     }//终结符的 First 集是本身终结符的 First 集是本身
185.     for (String s1 : terminal) {
186.         First.get(s1).add(s1);
187.     }
188.     int FirstSize = 0;
189.     do{
190.         FirstSize = 0;
191.         for (String s1 : First.keySet()) {
192.             FirstSize+=First.get(s1).size();
193.         }
194.         for (String[] strings : Symbol_Gram) {
195.             String lam = strings[1];

```

```

196.         String G = strings[0];
197.         setFirst(lam,G);
198.     }
199.     for (String s1 : First.keySet()) {
200.         FirstSize-=First.get(s1).size();
201.     }
202. }while (FirstSize != 0);
203. Follow.get(GStart).add("#");//文法开始符号 Follow 加入#
204.
205. int FollowSize = 0;
206. do{
207.     FollowSize = 0;
208.     for (String s1 : Follow.keySet()) {
209.         FollowSize+=Follow.get(s1).size();
210.     }
211.     for (String[] strings : Symbol_Gram) {
212.         String lam = strings[1];
213.         String G = strings[0];
214.         for (String s2 : noTerminal) {
215.             setFollow(lam,G,s2);
216.         }
217.     }
218.
219.
220.     for (String s1 : Follow.keySet()) {
221.         FollowSize-=Follow.get(s1).size();
222.     }
223. }while (FollowSize != 0);
224. for (String s1 : terminal) {
225.     Follow.remove(s1);
226. }
227. Map[] res = new Map[2];
228. res[0] = First;
229. res[1] = Follow;
230. return res;
231. }
232. private void setFirst(String lam,String G){
233.     String first = lam.substring(0,1);
234.     if(terminal.contains(first)){//终结符
235.         First.get(G).add(first);
236.     }
237.     else if(first.equals("ε")){//符号空
238.         First.get(G).add("ε");
239.     }

```

```

240.         else if(noTerminal.contains(first)){//非终结符
241.             First.get(G).addAll(First.get(first).stream().filter(S->!S.equals("ε")).collect(Collectors.toSet()));
242.             if(M(first,"ε")!=null){//是否可以推出空
243.                 setFirst(lam.substring(1),G);//扫描下一个
244.             }
245.         }
246.         else {
247.         }
248.     }
249.     private void setFollow(String lam,String G,String sym){//产生式 ->左边的符号 当先所求的非终结符
250.         if(!lam.contains(sym)){
251.             return;
252.         }
253.         int index = lam.indexOf(sym);//位置
254.         if(index == lam.length()-1){// 是\0
255.             Follow.get(sym).add("#");
256.             Follow.get(sym).addAll(Follow.get(G));//把产生式左边的 FOLLOW 加入到其的 FOLLOW 集中
257.             return;
258.         }
259.         else if(index < lam.length()-1){
260.             String next = lam.substring(index+1,index+2);//右边的符号
261.             if(terminal.contains(next)){//是终结符
262.                 Follow.get(sym).add(next);
263.             }
264.             else if(noTerminal.contains(next)){//非终结符
265.                 Follow.get(sym).addAll(First.get(next).stream().filter(S->!S.equals("ε")).collect(Collectors.toSet()));//把他的 First 集-ε 加入到当前分析的 Follow 集中
266.                 if(M(next,"ε")!=null){//检查可否推出空
267.                     //扫描下一个符号
268.                     StringBuffer changedLam = new StringBuffer(lam) ;
269.                     changedLam.deleteCharAt(index+1);//删除 达到左移的效果\
270.                     setFollow(changedLam.toString(),G,sym);
271.                 }
272.             }
273.
274.
275.         }
276.
277.     }
278.     private void setSelect(String[] strings,int index){

```

```

279.         if(index == strings[1].length()){//是空
280.             select.get(strings[0]+strings[1]).addAll(Follow.get(strings[0])
281.         );
282.         }
283.         else {
284.             String firstSym = strings[1].substring(index,index+1);
285.             if(terminal.contains(firstSym)){//如果是终结符
286.                 select.get(strings[0]+strings[1]).add(strings[1].substring(
287.                 0,1));
288.             }
289.             else if(firstSym.equals("ε")){//是空
290.                 select.get(strings[0]+strings[1]).addAll(Follow.get(strings
291.                 [0]));
292.             }
293.             else if(noTerminal.contains(firstSym)){//是非终结符
294.                 select.get(strings[0]+strings[1]).addAll(First.get(firstSym
295.                 ).stream().filter(S->!S.equals("ε")).collect(Collectors.toSet()));
296.                 if(M(firstSym,"ε") != null){//可以推空 则扫描下一个
297.                     setSelect(strings,index+1);
298.                 }
299.             }
300.         }
301.     }
302. }
303.
304. class MyStack{
305.     List<String> s;
306.     MyStack(){
307.         s = new LinkedList<>();
308.     }
309.     void push(String value){
310.         s.add(value);
311.     }
312.     void push(String...values){
313.         for(String value:values){
314.             push(value);
315.         }
316.     }
317.     String pop(){
318.         return s.remove(s.size()-1);
319.     }
320.     String getTop(){
321.         return s.get(s.size()-1);
322.     }
323.     @Override

```

```

319.     public String toString(){
320.         StringBuffer sb = new StringBuffer();
321.         for(String value:s){
322.             sb.append(value);
323.         }
324.         return sb.toString();
325.     }
326.     public Boolean isEmpty(){
327.         return s.size()==0;
328.     }
329. }

```

GUI.java

```

1. package 实验二__LL1 分析法;
2.
3. import javax.swing.*;
4. import javax.swing.table.AbstractTableModel;
5. import java.awt.*;
6. import java.awt.event.ActionEvent;
7. import java.awt.event.ActionListener;
8. import java.util.*;
9.
10. class Windows extends JFrame {
11.     JButton clear,confirm,setG,FF;
12.     JTextArea textArea;
13.     JTabbedPane tabbedPane;
14.     Solution sol;
15.     Windows(){
16.         setVisible(false);
17.         try{
18.             setIconImage(new ImageIcon("bilibili.PNG").getImage());
19.             Font f = new Font("Yahei Consolas Hybrid",Font.PLAIN,16);
20.             String names[]={ "MenuBar", "Menu", "MenuItem", "TextArea", "But
ton", "ScrollPane", "Table", "TabbedPane"};
21.             for (String item : names) {
22.                 UIManager.put(item+ ".font",f);
23.             }
24.             UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.Window
sLookAndFeel");
25.         }catch(Exception e){}
26.         init();

```

```

27.         setSize(800,600);//初始大小
28.         setLocation(300,200);//初始位置
29.         setVisible(true);//是否可视
30.         setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);//X 退出
31.     }
32.     void init(){
33.         setTitle("LL(1)分析法");
34.         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
35.         setVisible(true);
36.         setResizable(false);
37.         setLayout(null);
38.         sol = new Solution();
39.         initButton();
40.         initText();
41.         initResult();
42.
43.     }
44.     void initButton(){
45.         class clearListen implements ActionListener{
46.             @Override
47.             public void actionPerformed(ActionEvent e){
48.                 textArea.setText("");
49.                 tabbedPane.removeAll();
50.                 tabbedPane.updateUI();
51.             }
52.         }
53.         class confirmListen implements ActionListener{
54.             @Override
55.             public void actionPerformed(ActionEvent e){
56.                 tabbedPane.removeAll();
57.                 for(String text:textArea.getText().split("\n")){
58.                     if(text.length()<2 || !text.substring(text.length()-
59. 1).equals("#") ){
60.                         JOptionPane.showMessageDialog(null, "格式输入错误
61. ", "Error !", JOptionPane.ERROR_MESSAGE);
62.                         break;
63.                     }
64.                     addTable(text,sol.Solve(text));
65.                     tabbedPane.updateUI();
66.                 }
67.             }
68.         }

```

```

69.         class MyDialog extends JDialog implements ActionListener{
70.             JTextArea input;
71.             JButton confirm, cancel;
72.             String title;
73.             MyDialog(Windows f){
74.                 setLayout(null);
75.                 setResizable(false);
76.                 setIconImage(new ImageIcon("bilibili.PNG").getImage());
77.                 setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
78.                 setTitle("请输入语法");
79.                 input=new JTextArea("E -> TG \n" +
80.                                     "G -> +TG | -TG \n" +
81.                                     "G -> ε \n" +
82.                                     "T -> FS \n" +
83.                                     "S -> *FS | /FS \n" +
84.                                     "S -> ε \n" +
85.                                     "F -> (E) \n" +
86.                                     "F->i \n");
87.                 JScrollPane jScrollPane = new JScrollPane(input);
88.                 jScrollPane.setBounds(10,10,265,200);
89.                 add(jScrollPane);
90.
91.                 class confirmListener implements ActionListener{
92.                     @Override
93.                     public void actionPerformed(ActionEvent e){
94.                         String getInput = input.getText();
95.                         String setRes = sol.setG(getInput);
96.                         if(setRes!=null){
97.                             JOptionPane.showMessageDialog(null, setRes, "Err
or !", JOptionPane.ERROR_MESSAGE);
98.                         }
99.                         else {
100.                             setVisible(false);
101.                         }
102.                     }
103.                 }
104.                 confirm=new JButton("确定");
105.                 confirm.addActionListener(new confirmListener());
106.                 confirm.setBounds(195,220,80,30);
107.                 add(confirm);
108.
109.                 class cancelListener implements ActionListener{
110.                     @Override
111.                     public void actionPerformed(ActionEvent e){

```



```

112.             input.setText("E -> TG \n" +
113.                             "G -> +TG | -TG \n" +
114.                             "G -> ε \n" +
115.                             "T -> FS \n" +
116.                             "S -> *FS | /FS \n" +
117.                             "S -> ε \n" +
118.                             "F -> (E) \n" +
119.                             "F->i \n");
120.             setVisible(false);
121.         }
122.     }
123.     cancel=new JButton("取消");
124.     cancel.addActionListener(new cancellistener());
125.     cancel.setBounds(105,220,80,30);
126.     add(cancel);
127.
128.
129.     setBounds(600,260,300,300);
130.     setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
131. }
132. public void actionPerformed(ActionEvent e){
133.     setVisible(true);
134. }
135. }
136. class FFListen extends JDialog implements ActionListener{
137.     JTabbedPane jTabbedPane;
138.     FFListen(){
139.         jTabbedPane = new JTabbedPane();
140.         setLayout(null);
141.         setResizable(false);
142.         setIconImage(new ImageIcon("bilibili.PNG").getImage());
143.         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
144.         setTitle("Fist Follow 集");
145.         setBounds(650,300,300,400);
146.         jTabbedPane.setBounds(10,7,267,350);
147.         add(jTabbedPane);
148.
149.     }
150.     @Override
151.     public void actionPerformed(ActionEvent e){
152.         setVisible(true);
153.         jTabbedPane.removeAll();
154.         JTable First,Follow;
155.         Map<String,Set<String>>[] res = sol.getFF();

```

```
156.
157.         Object[][] FirstData = new Object[res[0].size()][2];
158.         int index = 0;
159.         for (String s : res[0].keySet()) {
160.             FirstData[index][0] = s;
161.             FirstData[index][1] = res[0].get(s).toString();
162.             index++;
163.         }
164.         Object[] columnNames = {"", ""};
165.         First = new JTable(FirstData, columnNames);
166.         First.setRowHeight(24);
167.         First.getTableHeader().setVisible(false);
168.         JScrollPane FirstScrollable = new JScrollPane(First);
169.         FirstScrollable.setBorder(null);
170.         jTabbedPane.addTab("First 集", FirstScrollable);
171.
172.         Object[][] FollowData = new Object[res[1].size()][2];
173.         index = 0;
174.         for (String s : res[1].keySet()) {
175.             FollowData[index][0] = s;
176.             FollowData[index][1] = res[1].get(s).toString();
177.             index++;
178.         }
179.         Follow = new JTable(FollowData, columnNames);
180.         Follow.setRowHeight(24);
181.         Follow.getTableHeader().setVisible(false);
182.         JScrollPane FollowScrollable = new JScrollPane(Follow);
183.         FollowScrollable.setBorder(null);
184.         jTabbedPane.addTab("Follow 集", FollowScrollable);
185.
186.     }
187. }
188.
189.
190.     clear = new JButton("清除");
191.     clear.setBounds(600, 160, 80, 30);
192.     clear.addActionListener(new clearListen());
193.
194.     confirm = new JButton("确认");
195.     confirm.setBounds(695, 160, 80, 30);
196.     confirm.addActionListener(new confirmListen());
197.
198.     setG = new JButton("自定义语法");
199.     setG.setBounds(460, 160, 120, 30);
```

```

200.         setG.addActionListener(new MyDialog(this));
201.
202.         FF = new JButton("Fist,Follow 集");
203.         FF.setBounds(260,160,180,30);
204.         FF.addActionListener(new FFListen());
205.
206.         class selectListen extends JDialog implements ActionListener{
207.             JTabbedPane jTabbedPane;
208.             selectListen(){
209.                 jTabbedPane = new JTabbedPane();
210.                 setLayout(null);
211.                 setResizable(false);
212.                 setIconImage(new ImageIcon("bilibili.PNG").getImage());
213.                 setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
214.                 setTitle("select 集");
215.                 setBounds(650,300,500,300);
216.                 jTabbedPane.setBounds(10,7,467,250);
217.                 add(jTabbedPane);
218.
219.             }
220.             @Override
221.             public void actionPerformed(ActionEvent e){
222.                 setVisible(true);
223.                 jTabbedPane.removeAll();
224.                 JTable First,Follow;
225.                 Map<String,Set<String>> res = sol.select;
226.
227.                 Object[][] FirstData = new Object[res.size()][2];
228.                 int index = 0;
229.                 for (String s : res.keySet()) {
230.                     FirstData[index][0] = s;
231.                     FirstData[index][1] = res.get(s).toString();
232.                     index++;
233.                 }
234.                 Object[] columnNames = {"", ""};
235.                 First = new JTable(FirstData, columnNames);
236.                 First.setRowHeight(24);
237.                 First.getTableHeader().setVisible(false);
238.                 JScrollPane FirstScrollable = new JScrollPane(First);
239.                 FirstScrollable.setBorder(null);
240.                 jTabbedPane.addTab("select 集",FirstScrollable);
241.
242.                 {
243.                     Map<String,Integer> Grammap = new HashMap<>();

```

```

244.         Map<String,Integer> endGmap = new HashMap<>();
245.
246.
247.         int a = 0;
248.         for (String s1 : sol.terminal) {
249.             //System.out.print("      "+s1);
250.             endGmap.put(s1,a);
251.             a++;
252.         }
253.         int b = 0;
254.         for (String s : sol.noTerminal) {
255.             Grammap.put(s,b);
256.             b++;
257.         }
258.         String[][] map = new String[Grammap.size()][endGmap.size()];
259.         //System.out.println();
260.         for (String s1 : sol.noTerminal) {
261.             //System.out.print(s1+": ");
262.             for (String s2 : sol.terminal) {
263.                 String resss = sol.AnaTable.get(s1+s2);
264.                 if(resss == null){
265.                     //System.out.print("      ");
266.                     map[Grammap.get(s1)][endGmap.get(s2)] = " "
267.                 }
268.                 else {
269.                     //System.out.print(resss+"      ");
270.                     map[Grammap.get(s1)][endGmap.get(s2)] = res
271.                 }
272.             }
273.             //System.out.println();
274.         }
275.         //System.out.println("=====");
276.
277.
278.
279.         //         for (String[] strings : map) {
280.         //             for (String string : strings) {
281.         //                 System.out.print(string+" ");
282.         //             }
283.         //             System.out.println();
284.         //         }

```

```
285.
286.
287.         String [] colum = new String[sol.noTerminal.size()+1];
288.         int counter = 0;
289.         for (String s : sol.noTerminal) {
290.             colum[counter] = s;
291.             counter++;
292.         }
293.         String[][] data = new String[map.length][map[0].length+
    1];
294.         for (int j = 0; j < data.length; j++ ) {
295.             data[j][0] = colum[j];
296.             for (int i = 1; i < data[0].length; i++) {
297.                 data[j][i] = map[j][i-1];
298.             }
299.         }
300.         String[] name = new String[sol.terminal.size()+1];
301.         int i = 1;
302.         name[0] = " ";
303.         for (String s : sol.terminal) {
304.             name[i] = s;
305.             i++;
306.         }
307.         JTable mmm = new JTable(data, name);
308.         mmm.setRowHeight(30);
309.         JScrollPane secsa = new JScrollPane(mmm);
310.         secsa.setBorder(null);
311.         jTabbedPane.addTab("分析表",secsa);
312.
313.     }
314.
315.
316.
317.
318.     }
319. }
320.
321.
322.     JButton select;
323.     select = new JButton("secect 集");
324.     select.setBounds(10,160,180,30);
325.     select.addActionListener(new selectListen());
326.     add(select);
```

```

327.
328.     add(clear);
329.     add(confirm);
330.     add(setG);
331.     add(FF);
332. }
333. void initText(){
334.     textArea = new JTextArea("i+i#\ni*i*i#\ni*(i+i)#\ni^i#");
335.     JScrollPane textAreaRollPane = new JScrollPane(textArea);
336.     textAreaRollPane.setBounds(10,10,765,140);
337.     add(textAreaRollPane);
338. }
339. void initResult(){
340.     tabbedPane = new JTabbedPane();
341.     tabbedPane.setBounds(10, 200, 765, 350);
342.     add(tabbedPane);
343. }
344. void addTable(String title,Vector vec ){
345.     TableDataModel tableDataModel = new TableDataModel(vec);
346.     JTable table = new JTable(tableDataModel);
347.     table.setVisible(true);
348.     table.setPreferredScrollableViewportSize(new Dimension(550, 100));
349.
350.     table.setRowHeight(24);
351.     JScrollPane tablePane = new JScrollPane(table);
352.     tablePane.setBounds(10, 200, 765, 350);
353.     tabbedPane.addTab(title,tablePane);
354. }
355.
356. class TableDataModel extends AbstractTableModel {
357.     private Vector<String[]> TableData;//用来存放表格数据的线性表
358.     private Vector<String> TableTitle;//表格的 列标题
359.     public TableDataModel(Vector data){
360.         String Names[] = {"步骤","分析栈","剩余输入栈","所用产生式","动作"};
361.         Vector Namessss = new Vector();
362.         for(String str:Names){
363.             Namessss.add(str);
364.         }
365.         TableTitle = Namessss;
366.         TableData = data;
367.     }
368.
369.     @Override

```

```
370.     public int getRowCount(){
371.         return TableData.size();
372.     }
373.     public int getColumnCount(){
374.         return TableTitle.size();
375.     }
376.     @Override
377.     public String getColumnName(int colum){
378.         return TableTitle.get(colum);
379.     }
380.     public Object getValueAt(int rowIndex, int columnIndex){
381.         String LineTemp[] = this.TableData.get(rowIndex);
382.         return LineTemp[columnIndex];
383.     }
384.     @Override
385.     public boolean isCellEditable(int rowIndex, int columnIndex){//不允许编辑
386.         return false;
387.     }
388. }
```

运行结果：

LL(1)分析法

i+i*
i*
i*(i+)
i^i

secect集

Fist,Follow集

自定义语法

清除

确认

i+i*
i*
i*(i+)
i^i

步骤	分析栈	剩余输入栈	所用产生式	动作
6	#G	+i* i* i*(i+) i^i	G -> +TG	POP,PUSH(GT+)
7	#GT+	+i* i* i*(i+) i^i		POP,GETNEXT(i)
8	#GT	i* i* i*(i+) i^i	T -> FS	POP,PUSH(SF)
9	#GSF	i* i* i*(i+) i^i	F -> i	POP,PUSH(i)
10	#GSi	i* i* i*(i+) i^i		POP,GETNEXT(i)
11	#GS	*i* i* i*(i+) i^i	S -> *FS	POP,PUSH(SF*)
12	#GSF*	*i* i* i*(i+) i^i		POP,GETNEXT(i)
13	#GSF	i* i* i*(i+) i^i	F -> i	POP,PUSH(i)
14	#GSi	i* i* i*(i+) i^i		POP,GETNEXT(i)
15	#GS	#	S -> ε	POP
16	#G	#	G -> ε	POP
17	#	#		

LL(1)分析法

i+i*
i*
i*(i+)
i^i

secect集

自定义语法

清除

确认

i+i*
i*
i*(i+)
i^i

步骤	分析栈	剩余输入栈	所用产生式	动作
6	#G	+i* i* i*(i+) i^i		POP,PUSH(GT+)
7	#GT+	+i* i* i*(i+) i^i		POP,GETNEXT(i)
8	#GT	i* i* i*(i+) i^i		POP,PUSH(SF)
9	#GSF	i* i* i*(i+) i^i	F -> i	POP,PUSH(i)
10	#GSi	i* i* i*(i+) i^i		POP,GETNEXT(i)
11	#GS	*i* i* i*(i+) i^i	S -> *FS	POP,PUSH(SF*)
12	#GSF*	*i* i* i*(i+) i^i		POP,GETNEXT(i)
13	#GSF	i* i* i*(i+) i^i	F -> i	POP,PUSH(i)
14	#GSi	i* i* i*(i+) i^i		POP,GETNEXT(i)
15	#GS	#	S -> ε	POP
16	#G	#	G -> ε	POP
17	#	#		

请输入语法

E -> TG
G -> +TG | -TG
G -> ε
T -> FS
S -> *FS | /FS
S -> ε
F -> (E)
F -> i

取消

确定

LL(1)分析法

$i+i\#$
 $i*i\#$
 $i*(i+i)\#$
 $i^\wedge i\#$

secect集 Fist, Fo

步骤	分析栈	剩余输入	动作
6	#G	$+i*i\#$	
7	#GT+	$+i*i\#$	
8	#GT	$i*i\#$	
9	#GSF	$i*i\#$	
10	#GSi	$i*i\#$	
11	#GS	$*i\#$	
12	#GSF*	$*i\#$	
13	#GSF	$i\#$	
14	#GSi	$i\#$	
15	#GS	#	$S \rightarrow \epsilon$ POP
16	#G	#	$G \rightarrow \epsilon$ POP
17	#	#	

Fist Follow 集

First集	Follow集
S	$[\epsilon, *, /]$
T	$[(, i]$
E	$[(, i]$
F	$[(, i]$
G	$[\epsilon, +, -]$
($[\epsilon]$
)	$[\epsilon]$
i	$[\epsilon]$
*	$[\epsilon]$
+	$[\epsilon]$
-	$[\epsilon]$
/	$[\epsilon]$

P,PUSH(GT+)
 P,GETNEXT(i)
 P,PUSH(SF)
 P,PUSH(i)
 P,GETNEXT(i)
 P,PUSH(SF*)
 P,GETNEXT(i)
 P,PUSH(i)
 POP,GETNEXT(i)

清除 确认

LL(1)分析法

$i+i\#$
 $i*i\#$
 $i*(i+i)\#$
 $i^\wedge i\#$

secect集 Fist, Fo

步骤	分析栈	剩余输入	动作
6	#G	$+i*i\#$	
7	#GT+	$+i*i\#$	
8	#GT	$i*i\#$	
9	#GSF	$i*i\#$	
10	#GSi	$i*i\#$	
11	#GS	$*i\#$	
12	#GSF*	$*i\#$	
13	#GSF	$i\#$	
14	#GSi	$i\#$	
15	#GS	#	$S \rightarrow \epsilon$ POP
16	#G	#	$G \rightarrow \epsilon$ POP
17	#	#	

Fist Follow 集

First集	Follow集
S	$[\#,), +, -]$
T	$[\#,), +, -]$
E	$[\#,)]$
F	$[\#,), *, +, -, /]$
G	$[\#,)]$

P,PUSH(GT+)
 P,GETNEXT(i)
 P,PUSH(SF)
 P,PUSH(i)
 P,GETNEXT(i)
 P,PUSH(SF*)
 P,GETNEXT(i)
 P,PUSH(i)
 POP,GETNEXT(i)

清除 确认

LL(1)分析法

i+i*
i*i*
i*(i+i)
i^i

select集

Fist, Fo

First Follow 集

select集 分析表

步骤	select集	分析表
6		
7	TFS	[(, i]
8	S*FS	[*]
9	Sε	[#, +, -]
10	Fi	[i]
11	F(E)	[(]
12	G+TG	[+]
13	ETG	[(, i]
14	G-TG	[-]
15		
16	#G	#
17	#	#

P, PUSH(GT+)
P, GETNEXT(i)
P, PUSH(SF)
P, PUSH(i)
P, GETNEXT(i)
P, PUSH(SF*)
P, GETNEXT(i)
P, PUSH(i)
POP, GETNEXT(i)
POP
POP

清除 确认

LL(1)分析法

i+i*
i*i*
i*(i+i)
i^i

select集

Fist, Fo

First Follow 集

select集 分析表

步骤	select集	分析表
6		
7	S	ε
8	T	FS
9	E	TG
10	F	(E)
11	G	ε
12		
13		
14		
15		
16	#G	#
17	#	#

P, PUSH(GT+)
P, GETNEXT(i)
P, PUSH(SF)
P, PUSH(i)
P, GETNEXT(i)
P, PUSH(SF*)
P, GETNEXT(i)
P, PUSH(i)
POP, GETNEXT(i)
POP
POP

清除 确认

4. 实验收获

本次实验设计的程序层次分明，程序界面与处理程序低耦合高内聚，分析器使用文法作为参数，对传入的字符串进行分析，并返回分析结果，此外通过调用类中的方法可以返回对应的 First 集 Follow 集等，便于显示。LL1 对分析法有了更好的理解和掌握。