

评阅项1计分＿＿＿＿＿＿

评阅项2计分＿＿＿＿＿＿

评阅项3计分＿＿＿＿＿＿

评阅项4计分＿＿＿＿＿＿

实验得分＿＿＿＿＿＿

评阅日期＿＿＿＿＿＿

**编译原理实验**

实验三：由底向上语法分析

实验选题：LR分析 构造识别活前缀的DFA

学生姓名： 屈 磊

学 号： 2016650313

专业班级： 16级计算机科学与技术2班

任课教师： 曹江莲

提交日期： 12.27

1. **实验目的**

通过本次实验，加深对移进规约分析法中算符优先法及LR(0)的理解，学习程序设计语言的语法分析器的手工编程方法。

1. **实验内容**

输入一个CFG，编程构造其识别活前缀的DFA，并判断是否有冲突。要求输出DFA的转换函数或状态转换图，以及LR(0)项目集规范簇。

1. **算法思想及问题对策**

1. 算法描述

1）改写文法：

对输入的文法，先进行拓广，然后逐条列出文法规则。

2）项目集的构造：

项目集存储在集合set结构内，对于给定的一个项目集，对其使用Closur（）函数求其闭包，具体是对集合内每条项目，若点.的后面是大写的非终结符A，则将文法规则中形如的A->.ȃ项目加入该项目集。直到项目集不再变化。

3）构造DFA:

本实验的DFA存在vector邻接表中，且只存有项目集标号和边上的字符，具体 的项目集通过项目集标号在map结构映射得到。构造DFA:首先构造项目集I0,然后从I0出发，深度搜索，生成对应的项目集，在搜索的过程中，若新生成的项目集为已有的项目集，则将其原来的项目集标号及边符号存入DFA,若新生成的项目集为新项目集，则把新标号及边符号存入DFA存入map同时继续深度搜索。直到DFA不在变化。

2．设计过程中的问题与对策

在构造DFA时，很关键的一步就是要判断生成的项目集是否已经存在了。

我通过构造镜像map map<int,set<string> >R用来读取项目集map<set<string>,int >R2 用来判断当前构造的项目集是否已经存在。二者在内容上只是把键值位置互换。

3. 本实验的特色

本实验对于DFA的存储采用了vector数组,且只存了项目集标号和边上的符号，具体的项目集存在map里。这样使得DFA存储更加方便。

1. **具体实现**
2. DFA存储结构定义声明, 镜像map的定义，文法的存储。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | struct Node{  int a;  char ch;  };  vector<Node> Dfa[20];  int cnt;  map<int,set<string> > R;  map<set<string>,int > R2;  vector<string> rule,rule2; //存储文法 |

1. 对输入的文法进行拓广

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | void genrenation(){ //拓广  string a,b;  a="H->"+firstch; //  rule2.push\_back(a);  for(int i=0;i<rule.size();i++){  int flag=0;  int h=0;  for(int j=3;j<rule[i].size();j++){  h++;  if(rule[i][j]=='|'){  flag=1;  b=rule[i][0];  a=b+"->"+rule[i].substr(j-h+1,h-1);  //cout<<a<<"\n";  rule2.push\_back(a);  h=0;  }    }  if(flag){  b=rule[i][0];  a=b+"->"+rule[i].substr(rule[i].size()-h,h);  rule2.push\_back(a);  //cout<<a<<"\n";  }else{  rule2.push\_back(rule[i]);  }  }  cout<<"拓广为文法G[H]:\n";  for(int i=0;i<rule2.size();i++){  string s=rule2[i];  cout<<s<<"\n";  }  cout<<endl;  } |

1. 构造项目集I0

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | void mI0(){ //构造I0  string a;  a="H->."+firstch;  I0.insert(a);  I0=f(firstch[0],I0);  R[0]=I0; cnt++; //将新集合存入 map  R2[I0]=0;  } |

1. 求集合的闭包

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | **set<string> Closure(set<string> a){** //求集合的闭包  **string s;**  **set<string> tmp=a;**  //先把集合取出来 避免读取加入元素后的集合  **set<string>::iterator it;**  **for(it=a.begin();it!=a.end();it++){**  **s=\*it;**  **for(int i=3;i<s.size()-1;i++){**  **if(s[i]=='.'&&s[i+1]>=65&&s[i+1]<=90){**  **tmp=f(s[i+1],tmp);**  **break;**  **}**    **}**  **}**  **return tmp;**  **}** |

1. 构造DFA

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | void mDfa(int a){ //构造Dfa  Node n;  string s;  set<string> tmp=R[a];  set<string>::iterator it;  for(it=tmp.begin();it!=tmp.end();it++){ //当前项目集项目里的.后有多少种字符就有多少下一态  set<string> tmp2;  s=\*it;  for(int i=3;i<s.size()-1;i++){  if(s[i]=='.'){  tmp2=shift(tmp,s[i+1]);  tmp2=Closure(tmp2);  //cout<<tmp2.size()<<endl;    if(R2.count(tmp2)){ //在map中已有这个集合  map<set<string>, int>::iterator it;  it=R2.find(tmp2);  n.a=it->second;  n.ch=s[i+1];  Dfa[a].push\_back(n);  //cout<<n.a<<" "<<n.ch<<endl;  }else{ //新的项目集  R[cnt]=tmp2;  R2[tmp2]=cnt;    n.a=cnt;  n.ch=s[i+1];  Dfa[a].push\_back(n);  cnt++;//必须先加1 否则下次进入还是原来cnt  mDfa(cnt-1);  }  break;  }  }  }  } |

1. **测试与运行情况**
2. 测试样例数据

本实验输入数据放在in3.txt中

程序中输入数据如下：

|  |  |
| --- | --- |
| 1  2  3 | 2  S->BB  B->aB|b |

1. 程序执行结果

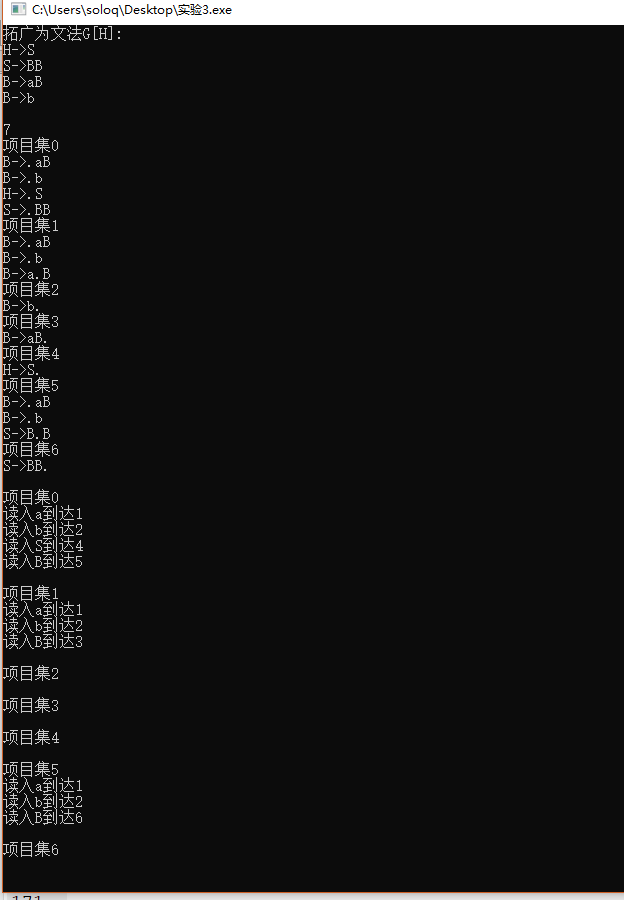


图2 程序测试结果

1. 程序执行结果分析

对于输入的文法，能够进行拓广，能够构造DFA，项目集存在map结构中。

1. **实验总结**

1.程序存在的问题

对于DFA的图结构还不能输出，只能将图存在vector邻接表中。

2. 收获

通过此次实验，我理解了自低向上分析中LR分析法。包括LR项目集的构造，字符串处理的细节，求项目集的闭包，还有最重要的生成DFA有向图。对我帮助很大。