# llin-黎辰

虽然弱，却执着







## 编译原理(三) 消除文法的左递归

标签： 编译原理

2015-11-28 08:45 1568人阅读 评论(0) 收藏 举报

分类： 编译原理 (9)



目录(?) [+]

# 算法的功能

对于任意上下文无关的文法消除左递归

# 问题分析

## 一、产生式直接消除左递归

形如$P \rightarrow Pa|\beta$可以通过直接消除转化为：

$$P \rightarrow \beta P'$$
$$P' \rightarrow aP'|\epsilon$$

## 二、产生式间接消除左递归

有时候虽然形式上产生式没有递归，但是因为形成了环，所以导致进行闭包运算后出现左递归，如下：

$$S \rightarrow Qc|c$$
$$Q \rightarrow Rb|b$$
$$R \rightarrow Sa|a$$

虽不具有左递归，但S、Q、R都是左递归的，因为经过若干次推导有

- SQcRbcSabc
- QRbSabQcab
- RSaQcaRbca

就显现出其左递归性了，这就是间接左递归文法。

***消除间接左递归的方法***是:

> 把间接左递归文法改写为直接左递归文法，然后用消除直接左递归的方法改写文法。
> 如果一个文法不含有回路，即形如PP的推导，也不含有以ε为右部的产生式，那么就可以采用下述**算法**消除文法的所有左递归。

消除左递归算法：

- （1）把文法G的所有非终结符按任一顺序排列，例如，A1，A2，…，An。
- （2）

```
for  (i=1; i<=n; i++)
   for  (j=1; j<=i-1; j++)
   { 把形如Ai→Ajγ的产生式改写成Ai→δ1γ /δ2γ /…/δkγ
       其中Aj→δ1 /δ2 /…/δk是关于的Aj全部规则;
       消除Ai规则中的直接左递归;
   }
```

- （3）化简由（2）所得到的文法，即去掉多余的规则。

利用此算法可以将上述文法进行改写，来消除左递归。
首先，令非终结符的排序为R、Q、S。对于R，不存在直接左递归。把R代入到Q中的相关规则中，则Q的规则变为Q→Sab/ ab/ b。
代换后的Q不含有直接左递归，将其代入S，S的规则变为S→Sabc/ abc/ bc/ c。
此时，S存在直接左递归。在消除了S的直接左递归后，得到整个文法为：

$$S \rightarrow abcS'|bcS'|cS'$$
$$S' \rightarrow abcS'|\varepsilon$$
$$Q \rightarrow Sab|ab|b$$
$$R \rightarrow Sa|a$$

可以看到从文法开始符号S出发，永远无法达到Q和R，所以关于Q和R的规则是多余的，将其删除并化简，最后得到文法G[S]为：

$$S \rightarrow abcS'|bcS'|cS'$$
$$S' \rightarrow abcS'|\varepsilon$$

当然如果对文法非终结符排序的不同，最后得到的文法在形式上可能不一样，但它们都是等价的。例如，如果对上述非终结符排序选为S、Q、R，那么最后得到的文法G[R]为：

$$R \rightarrow bcaR'|caR'|aR'$$
$$R' \rightarrow bcaR'|\varepsilon$$

容易证明上述两个文法是等价的。

# 代码实现

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <string>
#include <vector>
#include <queue>
#include <cctype>
#include <map>
#include <set>
#define MAX 507

using namespace std;

class WF
{
    public:
        string left;
        set<string> right;
        WF ( const string& temp )
        {
            left = temp;
            right.clear();
```

```
        }
        void print ( )
        {
            printf ( "%s::=" , left.c_str() );
            set<string>::iterator it = right.begin();
            printf ( "%s" , it->c_str());
            it++;
            for ( ; it!= right.end() ; it++ )
                printf ( "|%s" , it->c_str() );
            puts("");
        }
        void insert ( const string& temp )
        {
            right.insert(temp);
        }
};

map<string,int> VN_dic;
vector<WF> VN_set;
string start;
bool used[MAX];

//消除间接左递归
void remove1 ( )
{
    for ( int i = 0 ; i < VN_set.size() ; i++ )
        for ( int j = 0 ; j < i ; j++ )
        {
            vector<string> cont;
            set<string>& right1 = VN_set[i].right;
            set<string>& right2 = VN_set[j].right;
            char ch = VN_set[j].left[0];
            set<string>::iterator it1 = right1.begin();
            set<string>::iterator it2;
            for ( ; it1 != right1.end() ; it1++ )
                if ( it1->at(0) == ch )
                    for ( it2 = right2.begin() ; it2 != right2.end() ; it2++
                        cont.push_back ( *it2 + it1->substr(1) );
            int nn = right1.size();
            while ( nn-- )
            {
                if ( right1.begin()->at(0) == ch )
                    right1.erase ( right1.begin() );
                else
                {
                    cont.push_back ( *right1.begin() );
                    right1.erase ( right1.begin() );
                }
            }
            for ( int i = 0 ; i < cont.size() ; i++ )
                right1.insert ( cont[i] );
        }
#define DEBUG
#ifdef DEBUG
    for ( int i = 0 ; i < VN_set.size() ; i++ )
        VN_set[i].print();
#endif
}

//消除直接左递归
void remove2 ( )
{
    for ( int i = 0 ; i < VN_set.size() ; i++ )
    {
        char ch = VN_set[i].left[0];
        set<string>& temp = VN_set[i].right;
        set<string>::iterator it = temp.begin();
        string tt = VN_set[i].left.substr(0,1)+"\'";
        bool flag = true;
        for ( ; it != temp.end() ; it++ )
            if ( it->at(0) == ch )
```

阅读排行

```
95              {
96                  VN_set.push_back ( WF(tt));
97                  VN_dic[tt] = VN_set.size();
98                  flag = false;
99                  break;
100             }
101         int x = VN_dic[tt]-1;
102         if ( flag ) continue;
103         vector<string> cont;
104         set<string>& ss = VN_set[x].right;
105         ss.insert ( "~" );
106         while (!temp.empty())
107         {
108             if ( temp.begin()->at(0) == ch )
109                 ss.insert(temp.begin()->substr(1)+tt);
110             else
111             {
112                 //cout << "YES : " << temp.begin()->substr(1)+tt;
113                 cont.push_back (temp.begin()->substr(0)+tt);
114             }
115             temp.erase(temp.begin());
116         }
117         puts ("");
118         for ( int i = 0 ; i < cont.size() ; i++ )
119         {
120             //cout << cont[i] << endl;
121             temp.insert ( cont[i] );
122         }
123     }
124 #define DEBUG
125 #ifdef DEBUG
126     for ( int i = 0 ; i < VN_set.size() ; i++ )
127         VN_set[i].print();
128 #endif
129 }
130
131 void dfs ( int x )
132 {
133     if ( used[x] ) return;
134     used[x] = 1;
135     set<string>::iterator it = VN_set[x].right.begin();
136     for ( ; it != VN_set[x].right.end(); it++ )
137         for ( int i = 0 ; i < it->length() ; i++ )
138             if ( isupper(it->at(i)) )
139             {
140                 if ( it->length() > i+1 && it->at(i+1) == '\'' )
141                     dfs ( VN_dic[it->substr(i,2)]-1 );
142                 else
143                     dfs ( VN_dic[it->substr(i,1)]-1 );
144             }
145 }
146
147 //化简
148 void simplify ( )
149 {
150     memset ( used , 0 , sizeof ( used ) );
151     int x = VN_dic[start]-1;
152     dfs ( x );
153     puts ( "finish" );
154     vector<WF> res;
155     for ( int i = 0 ; i < VN_set.size() ; i++ )
156         if ( used[i] )
157             res.push_back ( VN_set[i] );
158     VN_set.clear();
159     VN_set = res;
160 }
161
162 void  print ()
163 {
164     puts("**********消除左递归后的结果********");
165     for ( int i = 0 ; i < VN_set.size() ; i++ )
```

Java设计模式(七) COR(j (6419)
Java设计模式(六) Comm (5369)
Java设计模式(五) Obser (5236)
Java 设计模式(十三) 接口 (4957)
Java设计模式(四) Facad (4124)
Java设计模式(二) Decor (3899)
Java PathFinder(一) Jav (3683)
Java设计模式(三) Visitor (3554)
Java 设计模式(八) Proxy (3149)
Java 设计模式(九) Strate (3030)

最新评论

编译原理(二) NFA的确定化及DF
g13121098278: 最后给不出起始节点和终止节点啊

编译原理(二) NFA的确定化及DF
g13121098278: 最后的minimize有注释吗 看不懂的感觉

关于最短路径算法的理解
WMY-2014: 算法功底真是强啊

编译原理(十) SLR文法分析法(算
cuixuange: good,很强

编译原理(四) 提取左因子法消除
黎辰: @msdnwolaile:如果有帮助的话，求关注^_^，可以一起交流学习~

编译原理(四) 提取左因子法消除
黎辰: @msdnwolaile:对于一个非终结符，First集是它通过产生式得到的句子可能的开头的字符的集...

编译原理(四) 提取左因子法消除
L未若: 博主，你好，能给个求FIRST的例子吗？？？，概念看的不是很懂，

Java 设计模式(十一) 里氏替换原
灿哥哥: 学习了

Java设计模式(五) Observer(观察
黎辰: @u010097777:这个系列我打算继续写下去，把自己对于一些实用的开源项目和设计模式结合起来理解...

Java设计模式(五) Observer(观察
u010097777: 让自己使用的工具应用在理论知识中，厉害。

```
166             VN_set[i].print();
167         puts("");
168     }
169
170     int main ( )
171     {
172         char buf[MAX];
173         int n;
174         VN_dic.clear();
175         VN_set.clear();
176         start="S";
177         puts ("请输入文法G[S]的产生式数量");
178         while ( ~scanf ("%d" , &n ) )
179         {
180             scanf ( "%d" , &n );
181             while ( n-- )
182             {
183                 scanf ( "%s" , buf );
184                 int len = strlen ( buf ),i;
185                 for ( i = 0 ; i < len ; i++ )
186                     if ( buf[i] == ':' )
187                     {
188                         buf[i] = 0;
189                         break;
190                     }
191                 string temp = buf;
192                 if ( !VN_dic[temp] )
193                 {
194                     VN_set.push_back ( WF(temp));
195                     VN_dic[temp] = VN_set.size();
196                 }
197                 int x = VN_dic[temp]-1;
198                 temp = buf+i+3;
199                 //cout <<"the right :  " << temp << endl;
200                 VN_set[x].insert(temp);
201             }
202             remove1();
203             remove2();
204             simplify();
205             print();
206             //puts ("请输入文法G[S]的产生式数量");
207         }
208     }
```

# 测试

**测试**样例：

```
6
R::=Sa
R::=a
Q::=Rb
Q::=b
S::=Qc
S::=c

```

测试结果:

```
请输入文法G[S]的产生式数量
R::=Sa|a
Q::=Sab|ab|b
S::=Sabc|abc|bc|c

R::=Sa|a
Q::=Sab|ab|b
S::=abcS'|bcS'|cS'
S'::=abcS'|~
finish
**********消除左递归后的结果********
S::=abcS'|bcS'|cS'
S'::=abcS'|~
```

上一篇 编译原理(二) NFA的确定化及DFA的最小化的算法及C++实现
下一篇 编译原理(四) 提取左因子法消除回溯



## 我的同类文章

### 编译原理（9）

- 编译原理(十) SLR文法分析法.. 2015-12-04 阅读 1081
- 编译原理(九) LR(0)文法分析... 2015-12-02 阅读 1791
- 编译原理(八) 算符优先分析... 2015-11-30 阅读 795
- 编译原理(七) 算符优先分析... 2015-11-29 阅读 [illegible]
- 编译原理(六) LL(1)文法分析... 2015-11-28 阅读 1101
- 编译原理(五) LL(1)文法分析... 2015-11-28 阅读 1595
- 编译原理(四) 提取左因子法... 2015-11-28 阅读 1214
- 编译原理(二) NFA的确定化及.. 2015-11-12 阅读 1561
- 编译原理(一) Chomsky文法... 2015-11-08 阅读 650



## 参考知识库


**软件测试知识库**

3743 关注 | 310 收录

**算法与数据结构知识库**

13631 关注 | 2320 收录

## 猜你在找

- C++ 单元测试（GoogleTest）
- C语言系列之 字符串相关算法
- QuickTest Professional深入剖析--【下部】
- Node.js进阶教程第四步：WebSocket
- 微服务场景下的自动化测试
- 编译原理 消除左递归
- 编译原理学习笔记二左递归消除递归下降
- 编译原理实验三语法分析递归下降法
- 编译原理学习笔记04孙悟空学72变之菩提老祖的阴谋可
- 求一道编译原理---清除左递归习题的解答



查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack

FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap