

llin-黎辰

虽然弱，却执着

目录视图

摘要视图

RSS 订阅

CSDN日报20170228——《技术晋升的评定与博弈》

程序员2月书讯

社区有奖问答--一起舞动酷炫的iOS动画

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)

版权声明：本文为博主原创文章，未经博主允许不得转载。

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：

$$\begin{aligned} P &\rightarrow \beta P' \\ P' &\rightarrow aP' | \epsilon \end{aligned}$$

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：

$$\begin{aligned} S &\rightarrow Qd|c \\ Q &\rightarrow Rb|b \\ R &\rightarrow Sa|a \end{aligned}$$

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：

个人资料



黎辰



访问： 348770次

积分： 11195

等级： BLOG > ?

排名： 第1082名

原创： 792篇

转载： 43篇

译文： 1篇

评论： 66条

文章搜索

博客专栏



Linux深入学习

文章： 11篇

阅读： 7025



编译原理算法实现

文章： 10篇

阅读： 14207



codeforces的概率专题

文章： 0篇

阅读： 0



可持久化数据结构专题

文章： 4篇

阅读： 2583

codeforces树上的专题



文章：3篇
阅读：2852



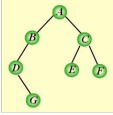
codeforces的数论专题
文章：45篇
阅读：23883



codeforces的图论专题
文章：3篇
阅读：2620



codeforces的dp专题
文章：78篇
阅读：52895



算法拙见
文章：12篇
阅读：6296



java程序设计深入学习
文章：33篇
阅读：70748

文章分类

- *ACM解题报告* (671)
- 动态规划----- (0)
- 动态规划 (206)
- 数位统计 (8)
- 区间dp (6)
- 记忆化搜索 (5)
- 最长上升子序列 (2)
- 最长公共子序列 (6)
- 背包 (21)
- 最大子段 (2)
- 最大完全矩形 (4)
- 环形dp (1)
- 树形dp (18)
- 概率dp (9)
- 状态压缩dp (5)
- rmq (8)
- 汉诺塔 (6)
- 数据结构----- (0)
- 数据结构 (38)
- 栈 (4)
- 单调栈 (2)
- 队列 (5)
- 优先队列 (9)
- 链表 (1)
- stl (13)
- 线段树 (31)
- 树套树型二维线段树 (2)
- 区间合并 (5)
- 树链剖分 (7)
- 划分树 (1)
- K-D 树 (4)
- 主席树 (4)
- 霍夫曼树 (2)
- 并查集 (18)
- 树状数组 (14)
- bitset (2)
- 数论----- (0)
- 数论 (94)
- 素数 (5)

- (1) 把文法G的所有非终结符按任一顺序排列，例如， A_1, A_2, \dots, A_n 。
- (2)

```
1  for (i=1; i<=n; i++)
2      for (j=1; j<=i-1; j++)
3          { 把形如 $A_i \rightarrow A_j \gamma$ 的产生式改写成 $A_i \rightarrow \delta_1 \gamma / \delta_2 \gamma / \dots / \delta_k \gamma$ 
4              其中 $A_j \rightarrow \delta_1 / \delta_2 / \dots / \delta_k$ 是关于的 $A_j$ 全部规则；
5              消除 $A_i$ 规则中的直接左递归；
6          }
```

- (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 $Q \rightarrow Sab / ab / b$ 。
代换后的Q不含有直接左递归，将其代入S，S的规则变为 $S \rightarrow Sabc / abc / bc / c$ 。
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：

$$\begin{aligned} S &\rightarrow abcS' | bcS' | cS' \\ S' &\rightarrow abcS' | \epsilon \\ Q &\rightarrow Sab | ab | b \\ R &\rightarrow Sa | a \end{aligned}$$

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：

$$\begin{aligned} S &\rightarrow abcS' | bcS' | cS' \\ S' &\rightarrow abcS' | \epsilon \end{aligned}$$

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：

$$\begin{aligned} R &\rightarrow bcaR' | caR' | aR' \\ R' &\rightarrow bcaR' | \epsilon \end{aligned}$$

容易证明上述两个文法是等价的。

代码实现

```
1  #include <iostream>
2  #include <cstdio>
3  #include <algorithm>
4  #include <cstring>
5  #include <string>
6  #include <vector>
7  #include <queue>
8  #include <cctype>
9  #include <map>
10 #include <set>
11 #define MAX 507
12
13 using namespace std;
14
15 class WF
16 {
17     public:
18         string left;
19         set<string> right;
20         WF ( const string& temp )
21         {
22             left = temp;
23             right.clear();
24         }
25     };
26
27 int main()
28 {
29     // ...
30 }
```

三角形数 (1)
欧拉定理 (12)
二次剩余--欧拉准则 (2)
鸽巢定理 (1)
同余定理 (12)
容斥定理 (5)
gcd (11)
中国剩余定理 (2)
拓展欧几里得 (4)
莫比乌斯反演 (4)
斐波那契数列 (6)
-----数学----- (0)
数学 (27)
概率 (14)
组合数学 (21)
Lucas定理 (4)
卡特兰数 (9)
第一类斯特林数 (0)
第二类斯特林数 (1)
全排列 (2)
矩阵快速幂 (13)
期望 (8)
全期望公式 (4)
指数分布 (1)
区域划分 (4)
高斯消元 (2)
物理题 (2)
母函数 (6)
整数拆分 (4)
五边形定理 (2)
矩阵乘法 (1)
等比数列快速求和 (1)
-----基本算法----- (0)
排序算法 (12)
贪心 (30)
离散化 (9)
二分 (28)
枚举 (32)
java (13)
三分 (1)
染色问题 (3)
归并排序 (1)
前缀和 (9)
状态压缩 (7)
极角排序 (1)
高精度 (5)
模拟 (28)
对数性质 (3)
递归 (4)
分治算法 (4)
CDQ分治 (1)
莫队算法 (1)
数制转换 (1)
分块算法 (1)
构造 (13)
位运算 (1)
-----博弈论----- (0)
博弈论 (26)
sg函数 (3)
局势段 (2)
-----字符串----- (0)
字符串 (31)
hash (4)
kmp (11)
拓展kmp (3)

```
24     }
25     void print ( )
26     {
27         printf ( "%s:=" , left.c_str() );
28         set<string>::iterator it = right.begin();
29         printf ( "%s" , it->c_str());
30         it++;
31         for ( ; it!= right.end() ; it++ )
32             printf ( "%s" , it->c_str() );
33         puts("");
34     }
35     void insert ( const string& temp )
36     {
37         right.insert(temp);
38     }
39 };
40
41 map<string,int> VN_dic;
42 vector<WF> VN_set;
43 string start;
44 bool used[MAX];
45
46 //消除间接左递归
47 void remove1 ( )
48 {
49     for ( int i = 0 ; i < VN_set.size() ; i++ )
50         for ( int j = 0 ; j < i ; j++ )
51         {
52             vector<string> cont;
53             set<string>& right1 = VN_set[i].right;
54             set<string>& right2 = VN_set[j].right;
55             char ch = VN_set[j].left[0];
56             set<string>::iterator it1 = right1.begin();
57             set<string>::iterator it2;
58             for ( ; it1 != right1.end() ; it1++ )
59                 if ( it1->at(0) == ch )
60                     for ( it2 = right2.begin() ; it2 != right2.end() ; it2++ )
61                         cont.push_back ( *it2 + it1->substr(1) );
62             int nn = right1.size();
63             while ( nn-- )
64             {
65                 if ( right1.begin()->at(0) == ch )
66                     right1.erase ( right1.begin() );
67                 else
68                 {
69                     cont.push_back ( *right1.begin() );
70                     right1.erase ( right1.begin() );
71                 }
72             }
73             for ( int i = 0 ; i < cont.size() ; i++ )
74                 right1.insert ( cont[i] );
75         }
76 #define DEBUG
77 #ifdef DEBUG
78     for ( int i = 0 ; i < VN_set.size() ; i++ )
79         VN_set[i].print();
80 #endif
81 }
82
83 //消除直接左递归
84 void remove2 ( )
85 {
86     for ( int i = 0 ; i < VN_set.size() ; i++ )
87     {
88         char ch = VN_set[i].left[0];
89         set<string>& temp = VN_set[i].right;
90         set<string>::iterator it = temp.begin();
91         string tt = VN_set[i].left.substr(0,1)+"'";
92         bool flag = true;
93         for ( ; it != temp.end() ; it++ )
94             if ( it->at(0) == ch )
```

trie (6)

manacher算法 (7)

后缀数组 (5)

AC自动机 (8)

后缀自动机 (1)

-----搜索----- (0)

搜索 (22)

bfs (19)

dfs (30)

奇偶性剪枝 (1)

可行性剪枝 (2)

最优化剪枝 (1)

迭代加深搜索 (1)

-----图论----- (0)

图论 (29)

LCA (8)

约瑟夫环 (1)

拓扑排序 (3)

树 (5)

最小生成树--prim (5)

最小生成树--kruskal (4)

最大伪森林子图 (1)

最短路 (14)

最短路--spfa (5)

最短路--dijkstra (2)

差分约束系统 (1)

最短路--floyd (3)

传递闭包--floyd (1)

二分图最大匹配--匈牙利算法 (9)

二分图最大匹配--最大独立点集 (3)

二分图最大匹配--最小点覆盖 (4)

二分图最大匹配--最小路径覆盖 (2)

割点--tarjan (4)

割边--tarjan (4)

强连通分量--tarjan (4)

点双连通分量--tarjan (3)

边双连通分量--tarjan (2)

2-sat--最小字典序解 (2)

2-sat--tarjan (5)

-----计算几何----- (0)

计算几何 (19)

解析几何 (2)

凸包--Graham-Scan (1)

凸包--卷包裹法 (2)

最近相邻点对 (1)

-----概率论----- (0)

随机事件和概率 (6)

-----网络流----- (0)

最大流--dinic (4)

最小割--dinic (1)

二分图最小点权覆盖集--dinic (1)

二分图最大点权独立集--dinic (1)

全局最小割--Stoer-Wagner (2)

最小费用最大流--spfa (2)

---java程序设计深入学习--- (0)

java---i/o (4)

java---collections (7)

操作系统 (11)

图像处理 (2)

可视化 (0)

编译原理 (10)

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(i (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 and 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：

```
1 6
2 R::=Sa
3 R::=a
4 Q::=Rb
5 Q::=b
6 S::=Qc
7 S::=c
8
```

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'|~
```

顶 踩
0 0

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 1001
- 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



阿里云，30+产品免费使用

全球领先、安全、稳定的云计算产品。马上注册，即享受30款产品免费套餐。

广告

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72变之菩提老祖的阴谋可求一道编译原理——消除左递归习题的解答

清理我的 MAC

MacKeeper - 即刻清理 Mac。确保 Mac 安全。



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 | | | | | |

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服

杂志客服

微博客服

webmaster@csdn.net

400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |



单身公寓



创意产品设计



卖狗网



德云社门票

© 1999-2016, CSDN.NET, All Rights Reserved



□