中山大学数据科学与计算机学院本科生实验报告

课程名称：编译器构造实验 任课教师：陈炬桦

|  |  |  |  |
| --- | --- | --- | --- |
| 学年度 | 2018-2019 | 学期 | 第二学期 |
| 年级 | 16级 | 专业（方向） | 计算机科学与技术 |
| 学号 | 16337341 | 姓名 | 朱志儒 |
| 电话 | 15989184223 | Email | 739741104@qq.com |
| 开始日期 | 2019/3/15 | 完成日期 | 2019/3/15 |

1. 实验题目

**输入文法压缩自产生式文法和不可达文法**

* 1. Description

输入开始符号，非终结符，终结符，产生式

压缩自产生式文法和不可达文法后，按非终结符顺序输出产生式；

* 1. Input

输入开始符号；

非终结符个数，非终结符，空格符分隔；

终结符个数，终结符，空格符分隔；

产生式的个数，各产生式的左边和右边符号，空格符分隔；

* 1. Output

delete self production:自产生式文法

unreached Vn:不可达非终结符

delete production:不可达产生式

delete VN:不可达非终结符

G[开始符号]：

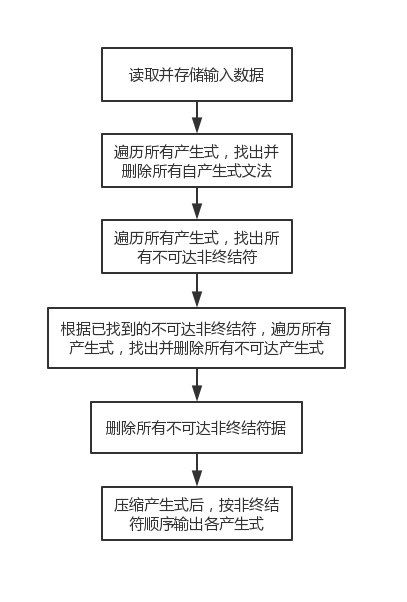
压缩自产生式文法和不可达文法后，按非终结符顺序输出各产生式；

1. 算法描述（介绍程序模块功能，流程图）

算法：

* 读取并存储输入数据
* 遍历所有产生式，找出并删除所有自产生式文法
* 遍历所有产生式，找出所有不可达非终结符
* 根据已找到的不可达非终结符，遍历所有产生式，找出并删除所有不可达产生式
* 删除所有不可达非终结符
* 压缩产生式后，按非终结符顺序输出各产生式

流程图：



1. 测试数据（2组）

|  |  |  |
| --- | --- | --- |
|  | 输入 | 输出 |
| 第一组 | Z  8 Z E F P G T Q S  3 + \* i  18  Z E+T  E E  P G  F F  P G  G G  T T\*i  Q E  S i  E S+F  F FP  G GG  Q E+F  E T  F P  G F  Q T  Q S | delete self production:E::=E  delete self production:F::=F  delete self production:G::=G  unreached Vn:Q  delete production:Q::=E  delete production:Q::=E+F  delete production:Q::=T  delete production:Q::=S  delete VN:Q  G[Z]:  Z::=E+T  E::=S+F | T  F::=FP | P  P::=G | G  G::=GG | F  T::=T\*i  S::=i |
| 第二组 | E  3 E T F H  5 + \* ( ) i  6  E E+T  E E  H F  T T\*F  E T  T F  F (E)  F i | delete self production:E::=E  unreached Vn:H  delete production:H::=F  delete VN:H  G[E]:  E::=E+T | T  T::=T\*F | F  F::=(E) | i |

1. 程序清单
2. #include <iostream>
3. #include <string>
4. #include <algorithm>
5. #include <vector>
6. **using** **namespace** std;
8. **struct** VN {
9. **int** nv;
10. string vn[100];
11. } vns;
13. **struct** VT {
14. **int** nt;
15. string vt[100];
16. } vts;
18. **struct** PS {
19. **int** np;
20. string pl[100], pr[100];
21. } ps;
23. string s;
25. **bool** reachable(PS source, string vn) {
26. **for** (**int** i = 0; i < source.np; ++i)
27. **if** (source.pr[i].find(vn) != string::npos)
28. **return** **true**;
29. **return** **false**;
30. }
32. **bool** is\_unreached(VN unreached, string vn) {
33. **for** (**int** i = 0; i < unreached.nv; ++i)
34. **if** (unreached.vn[i] == vn)
35. **return** **true**;
36. **return** **false**;
37. }
39. **int** main() {
40. cin >> s >> vns.nv;
41. **for** (**int** i = 0; i < vns.nv; ++i)
42. cin >> vns.vn[i];
43. cin >> vts.nt;
44. **for** (**int** i = 0; i < vts.nt; ++i)
45. cin >> vts.vt[i];
46. cin >> ps.np;
47. **for** (**int** i = 0; i < ps.np; ++i)
48. cin >> ps.pl[i] >> ps.pr[i];
49. PS update = { 0 };
50. **for** (**int** i = 0; i < ps.np; ++i) {
51. **if** (ps.pl[i] == ps.pr[i]) {
52. cout << "delete self production:" << ps.pl[i] << "::=" << ps.pr[i] << endl;
53. }
54. **else** {
55. update.pl[update.np] = ps.pl[i];
56. update.pr[update.np] = ps.pr[i];
57. update.np++;
58. }
59. }
60. VN unreached = { 0 };
61. **for** (**int** i = 0; i < vns.nv; ++i) {
62. **if** (vns.vn[i] != s && !reachable(update, vns.vn[i])) {
63. unreached.vn[unreached.nv++] = vns.vn[i];
64. cout << "unreached Vn:" << vns.vn[i] << endl;
65. }
66. }
67. PS final = { 0 }, output = { 0 };
68. **for** (**int** i = 0; i < unreached.nv; ++i) {
69. **for** (**int** j = 0; j < update.np; ++j) {
70. **if** (update.pl[j] == unreached.vn[i]) {
71. cout << "delete production:" << update.pl[j] << "::=" << update.pr[j] << endl;
72. }
73. }
74. }
75. **for** (**int** i = 0; i < unreached.nv; ++i) {
76. cout << "delete VN:" << unreached.vn[i] << endl;
77. }
78. **for** (**int** i = 0; i < update.np; ++i) {
79. **if** (!is\_unreached(unreached, update.pl[i])) {
80. final.pl[final.np] = update.pl[i];
81. final.pr[final.np] = update.pr[i];
82. final.np++;
83. }
84. }
85. vector<string> tmp;
86. **for** (**int** i = 0; i < final.np; ++i) {
87. **if** (find(tmp.begin(), tmp.end(), final.pl[i]) == tmp.end()) {
88. tmp.push\_back(final.pl[i]);
89. output.pl[output.np] = final.pl[i];
90. output.pr[output.np] = final.pr[i];
91. **for** (**int** j = i + 1; j < final.np; ++j) {
92. **if** (final.pl[i] == final.pl[j]) {
93. output.pr[output.np] += " | " + final.pr[j];
94. }
95. }
96. output.np++;
97. }
98. }
99. cout << "G[" << s << "]:" << endl;
100. **for** (**int** i = 0; i < vns.nv; ++i) {
101. **if** (!is\_unreached(unreached, vns.vn[i])) {
102. **for** (**int** j = 0; j < output.np; ++j) {
103. **if** (output.pl[j] == vns.vn[i]) {
104. cout << output.pl[j] << "::=" << output.pr[j] << endl;
105. **break**;
106. }
107. }
108. }
109. }
110. **return** 0;
111. }