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 教学助理（TA）：

|  |  |  |  |
| --- | --- | --- | --- |
| 学年度 | 2018-2019 | 学期 | 第二学期 |
| 年级 | 2016级 | 专业（方向） | 计算机科学与技术 |
| 学号 | 16337233 | 姓名 | 王凯祺 |
| 电话 | 13434386182 | Email | [Wangkq3@mail2.sysu.edu.cn](mailto:Wangkq3@mail2.sysu.edu.cn) |
| 开始日期 | 2019/3/22 | 完成日期 | 2019/3/22 |

1. 实验题目

**εNDFA 确定化**

**1.1 Description**

输入字母表，非确定εNDFA 确定化ＦＡ状态集，映射集；用造表法算法。  
输出确定ＦＡ状态集，映射集；

**1.2 Input**

 输入字母个数，字母表  
状态个数，状态表（状态名称，开始状态，终止状态：0否1是），空格符分隔；  
映射个数，映射表（起，终，字母），空格符分隔，k表示ε；

**1.3 Output**

Determine State:  
状态表: 状态名称，开始状态，终止状态，[子集]  
Determine Mapping:  
映射表: 起，终，字母,(顺序号

2. 算法描述(介绍程序模块功能；流程图)

(1) 将输入的状态保存在节点上，输入的映射保存在边（矩阵）上。

(2) 对矩阵做一次求闭包的floyd算法，用于处理以下3类情况：a) A🡪B（转移字符为空）, B🡪C（转移字符为空），则必有A🡪C（转移字符为空）；b) A🡪B（转移字符为x/y），B🡪C（转移字符为空），则必有A🡪C（转移字符为x/y）；c) A🡪B（转移字符为空），B🡪C（转移字符为x/y），则必有A🡪C（转移字符为x/y）。

(3) 从初始状态（同时记为DFA的初始状态）开始，对原NDFA进行广度优先搜索（BFS）。每次从队列中取出一个DFA状态（实际为NDFA的状态集合），枚举字母，再枚举NDFA状态集合的每个元素，记录在当前状态集合下，每种字母可以到达的状态集合。把能到达的状态集合视为DFA的一个状态，连边。

(4) BFS结束后，DFA的状态和连边全部确定。

3. 测试数据

样例输入

2

x y

9

0 1 0

1 0 0

2 0 0

3 0 0

4 0 0

5 0 0

6 0 0

7 0 0

8 0 1

12

0 1 x

0 3 y

0 6 x

1 2 k

2 2 y

2 8 k

3 4 k

4 4 x

4 5 k

5 8 y

6 7 y

7 8 x

样例输出

Determine State:

0 1 0 [0]

1 0 1 [1268]

2 0 0 [345]

3 0 1 [278]

4 0 0 [45]

5 0 1 [8]

6 0 1 [28]

Determine Mapping:

0 1 x (0

0 2 y (1

1 3 y (2

2 4 x (3

2 5 y (4

3 5 x (5

3 6 y (6

4 4 x (7

4 5 y (8

6 6 y (9

4. 程序清单

#include <bits/stdc++.h>

#define N 100

#define C 3

using namespace std;

int c, n, m, cnt, S[N], T[N], g[N][N][C];

int ex[N], ey[N], ew[N];

int nS[N], nT[N], mask[N];

char alpha[C];

map<int, int> mp;

queue<int> q;

bool vis(int x) {

if (mp.find(x) != mp.end()) return 1;

return 0;

}

int mapv(int x) {

if (mp.find(x) != mp.end()) return mp[x];

else {

for (int i = 0; i < n; ++i)

if (x >> i & 1) {

if (S[i]) nS[cnt] = 1;

if (T[i]) nT[cnt] = 1;

}

mask[cnt] = x;

mp[x] = cnt++;

return mp[x];

}

}

int main() {

scanf("%d", &c);

for (int i = 0; i < c; ++i) {

char st[2];

scanf("%s", st);

alpha[i] = st[0];

}

alpha[c++] = 'k';

scanf("%d", &n);

for (int i = 0; i < n; ++i) {

int id;

scanf("%d%d%d", &id, &S[i], &T[i]);

assert(i == id);

}

scanf("%d", &m);

for (int i = 0; i < m; ++i) {

int x, y, w = -1;

char st[2];

scanf("%d%d%s", &x, &y, st);

for (int j = 0; j < c; ++j)

if (st[0] == alpha[j]) {

w = j;

break;

}

g[x][y][w] = 1;

}

for (int k = 0; k < n; ++k)

for (int i = 0; i < n; ++i)

for (int j = 0; j < n; ++j) {

if (g[i][k][2] && g[k][j][2]) g[i][j][2] = 1;

if (g[i][k][2] && g[k][j][0]) g[i][j][0] = 1;

if (g[i][k][2] && g[k][j][1]) g[i][j][1] = 1;

if (g[i][k][0] && g[k][j][2]) g[i][j][0] = 1;

if (g[i][k][1] && g[k][j][2]) g[i][j][1] = 1;

}

int start = 0;

for (int i = 0; i < n; ++i)

if (S[i])

start |= 1 << i;

cnt = 0;

while (!q.empty()) q.pop();

q.push(mapv(start));

int eM = 0;

while (!q.empty()) {

int x = q.front();

q.pop();

for (int a = 0; a < c - 1; ++a) {

int state = 0;

for (int xx = 0; xx < n; ++xx)

if (mask[x] >> xx & 1)

for (int y = 0; y < n; ++y)

if (g[xx][y][a]) state |= 1 << y;

if (state == 0) continue;

if (!vis(state)) {

q.push(mapv(state));

}

++ eM;

ex[eM] = x;

ey[eM] = mapv(state);

ew[eM] = a;

}

}

puts("Determine State:");

for (int i = 0; i < cnt; ++i) {

printf("%d %d %d [", i, nS[i], nT[i]);

for (int j = 0; j < n; ++j)

if (mask[i] >> j & 1)

printf("%d", j);

printf("]\n");

}

puts("Determine Mapping:");

for (int i = 1; i <= eM; ++i) {

printf("%d %d %c (%d\n", ex[i], ey[i], alpha[ew[i]], i - 1);

}

}