**华北水利水电大学 形式语言与自动机 实验报告**

**2022-2023 学年 第**  一 **学期 2020 级 人工智能 专业**

**指导老师 张瑞霞 学号 202018526 姓名 高树林**

## 实验二 DFA识别句子

1. 实验目的

（1）掌握DFA的形式化表示。

（2）掌握DFA识别句子的基本原理及过程。

1. 进一步理解用模型解决实际问题的基本过程。
2. 实验内容

设计并实现DFA，实现对输入任意句子的识别。若不是DFA可识别的句子，则给出句子不可识别的原因；若是识别句子，可给出识别句子的过程。

1. 实验指导

（1）设计固定DFA。也就是说用if-then-else（一般用来实现字母表中只有两个字母的情况）、switch（大于两个字母的情况）、for（用于控制输入字符串，长度为n的字符串，for循环n次）等语句表示DFA。一个函数定义一个DFA；

（2）设计文件形式存储DFA。设计文件格式，可以先检查DFA形式化的合法性，再动态生产DFA；设计合理的数据结构来表示DFA，可以采用图结构、可以采用数组表示分量方式，其中状态转移函数的表示是关键。

（3）关于DFA识别句子过程。采用不同的数据结构，识别过程略有不同，但总体思路是根据输入句子自动实现状态转移函数的跳转。

（4）图形化表示。用所使用语言中图形功能实现图形化的dfa。（选作）

1. 实验设计与实现

包括主要设计思想、主要数据结构、主要程序设计流程图或伪代码、主要程序代码。

**设计思想：**在设计DFA的代码实现的过程中，首先要将自动机的五元组信息输入到程序中。之后抽取最为重要的状态函数部分的内容，将其整理成一个如下所示的映射表。

|  |  |
| --- | --- |
| （当前点，读取的状态） | 下一个点 |

这个结构可以用字典的数据结构来实现，即（当前点，读取的状态）作为键，下一个点作为值，每一次读取就是一次迭代更新当前节点的过程。当迭代结束后，如果当前点是终极符，那么就说明该句子是能够被自动机所识别的，否则就是该自动机所不能识别的状态。

**数据结构：**利用python内置的字典结构。用字典结构的好处在于：可以通过对键值对的索引达到迭代更新，大大降低了程序的复杂程度。同时，字典的好处还在于将节点以及节点读取状态的关系存储在当度的一个数据结构中，稳定性强。

**流程图：**

输入自动机五元组的内容

可被识别接受

写入文件

循环遍历，迭代更新状态

抽取状态转移函数，构建映射表

输入句子

不能被识别接受

图1 DFA识别句子并记录识别过程的流程图

**代码：**

1. **def** DFA(beg, end):
2. next = beg
3. present = next
4. **for** i **in** string:
5. next = rule['(%s,%s)' % (present, i)]
6. # result.append('(%s,%s)=%s' % (present, i, next))
7. f.write('(%s,%s)=%s' % (present, i, next) + '\n')
8. present = next
9. **if** present == end:
10. **return** 0
11. **return** 1

14. **if** \_\_name\_\_ == '\_\_main\_\_':
15. # F = 'q0'  # input('输入起始符：')
16. # E = 'q3'  # input('输入终态符：')
17. result = []
18. # rule = {'(q0,0)': 'q1', '(q0,1)': 'q0', '(q1,0)': 'q2', '(q1,1)': 'q0', '(q2,0)': 'q3', '(q2,1)': 'q0',
19. #         '(q3,0)': 'q3',
20. #         '(q3,1)': 'q0'}
21. # print(rule)
23. **print**('输入状态转移函数，按“#”键结束')
24. rule1 = []
25. **while** 1:
26. a = input()
27. **if** a == "#":
28. **break**
29. rule1.append(a)
30. **if** '' **in** rule1:
31. **del** rule1[rule1.index('')]
32. # print(rule1)
33. rule = {i.split('=')[0]: i.split('=')[-1] **for** i **in** rule1}
34. name = input('输入存储文件的名称：')
35. F = input('输入起始符：')
36. E = input('输入终态符：')
37. string = input('输入句子:')
38. f = open(name + '.txt', 'w', encoding='utf-8')
39. f.write('Q=%s' % str(sorted(list(set((rule.values()))))).replace('[', '{').replace(']', '}') + '\n')  # 写第一行，状态集合
40. f.write(
41. 'A=%s' % str(sorted(list(set([int(i[1:-1].split(',')[-1]) **for** i **in** rule.keys()])))).replace('[', '{').replace(
42. ']', '}') + '\n')  # 写第二行，字母表
43. f.write('T:' + '\n')  # 写第三行
44. **for** i **in** rule1:
45. f.write("%s\n" % i)
47. f.write('输入的句子是%s，识别过程为：\n' % (string))
48. judge = ['', '不']
49. # DFA(F, E)
50. f.write('%s状态%s可以被该DFA接受' % (string, judge[DFA(F, E)]))
51. **print**('Finished!')

**代码及流程说明：**

代码按照流程图的第一步开始，首先要求用户输入DFA的数学表示。这个输入时分开输入的，首先要求输入的是状态转移函数，因为状态转移函数时DFA的核心内容，有了它就能够直到分别存在哪些状态和字母表中的元素。有了这些还不够，需要给定自动机的起始符和终态符，否则自动机不能正常工作。在输入状态转移函数的过程中，用一个rule1函数将每一个状态函数都加进列表里面，在输入和结束后在经过统一处理，将状态函数改编为字典映射关系，其原理是：将当前节点和即将读取的状态集成在一个元组里面成为一个整体，这个元组所映射的就是当前状态的下一个状态。之后将这些信息全部写到文本文档里面，写如的时候需要调用python自带的open函数，在这里一定要注意将encoding改为utf-8的形式，原因在六实验总结中会提到。当然，这里面有一个文件名的参数，因此在写入之前要先定义一个文件夹的名称。

在进入·迭代函数时，需要传递开始符和终极符两个参数，最开始的时候当前节点是空的，而当前节点的下一个节点是第一个节点，也就是开始符。将下一个节点赋给当前节点，那么当前节点就是开始符。之后进入循环，首先找的就是下一个状态next。next的查找方式就是按照前文所说利用当前节点和将要读取的状态集成组合的元组为索引找到下一个结点，并将这个过程写进文件里，之后更新当前结点的值，依次读取string的每一的字符（句子的状态）。当遍历结束（句子的每一个状态都被接受），需要判断当前结点是不是终结符，如果是，则表明该句子能够被DFA所接受与识别，若不是终结符，表明该句子不能够被DFA所接受与识别。最后把能否识别的结果写进文件里面，这里做了一个小技巧，在函数中，不能识别返回1，能识别返回0，这两个值可以作为能判断否识别这个列表索引。

1. 实验结果及分析

本次实验做了两组实验，分别是状态集是两个和三个的情况。每一组实验又各做了一个对比试验，共四组实验。下面是四组实验的结果。

**结果1：**当字母表中含有两个状态（0，1）时，验证自动机识别以000结尾的句子正例结果如下图2所示。

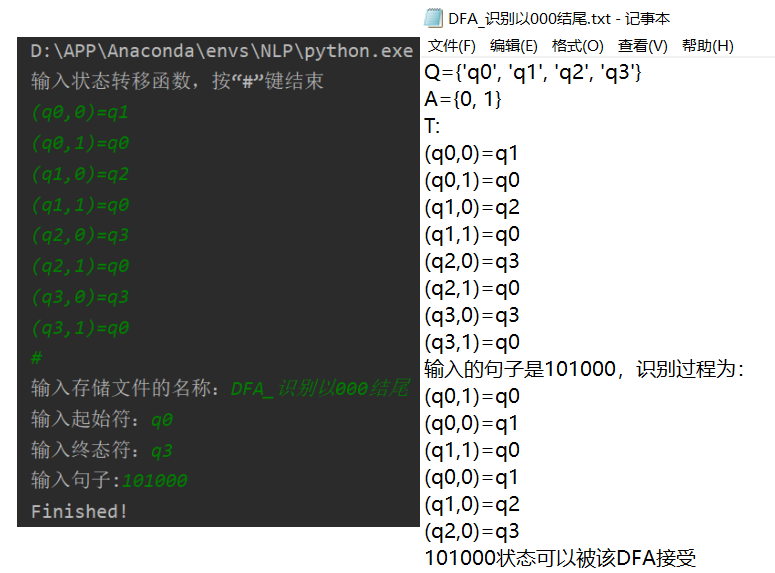


图2 成功识别以000结尾的句子

**结果2：**当字母表中含有两个状态（0，1）时，验证自动机识别以000结尾的句子反例结果如下图3所示。

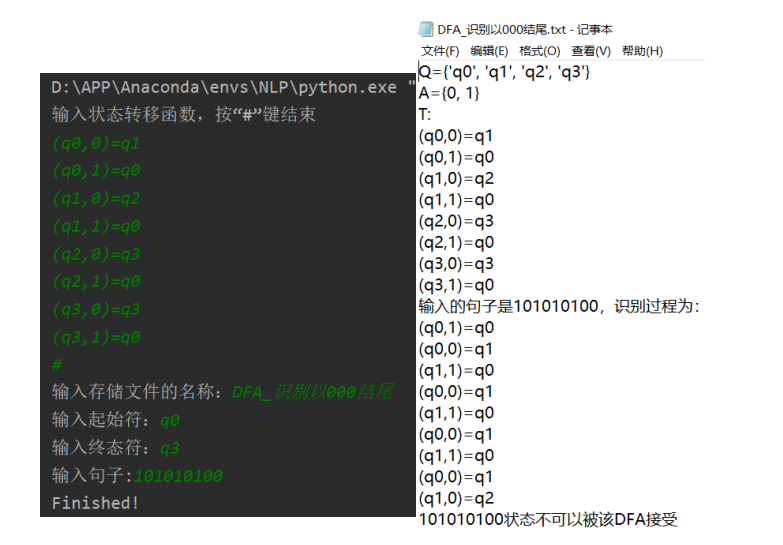


图3识别失败以000结尾的句子

**结果3：**当字母表中含有三个状态（0，1，2）时，验证自动机识别含有012的句子正例结果如下图4所示。

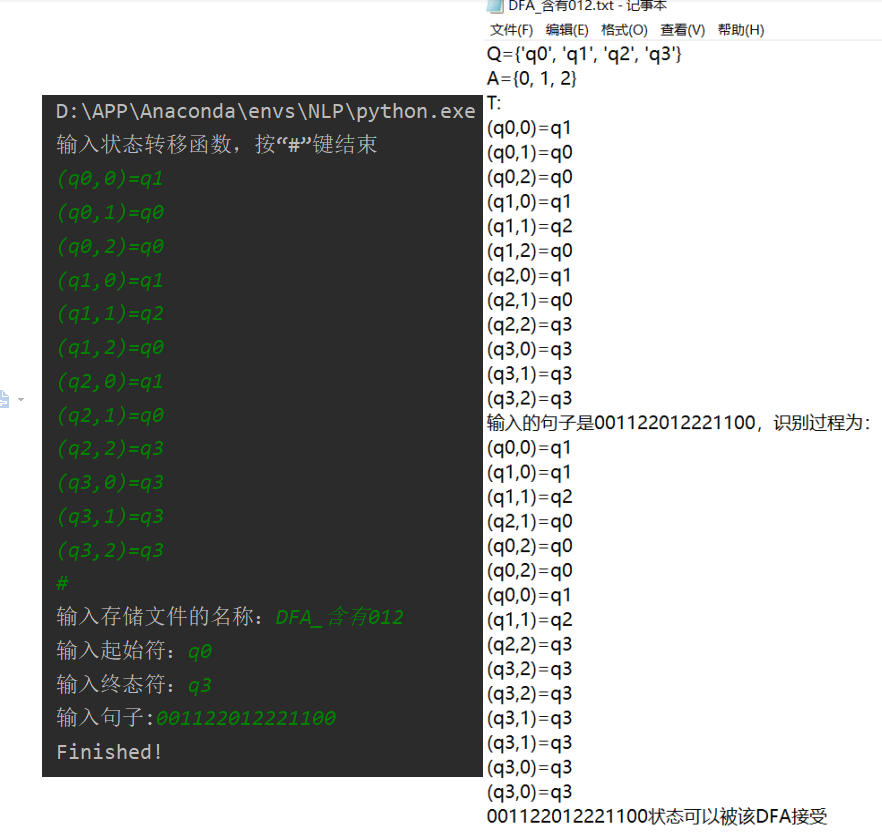


图4 成功识别含012的句子

**结果4：**当字母表中含有三个状态（0，1，2）时，验证自动机识别含有012的句子反例结果如下图4所示。

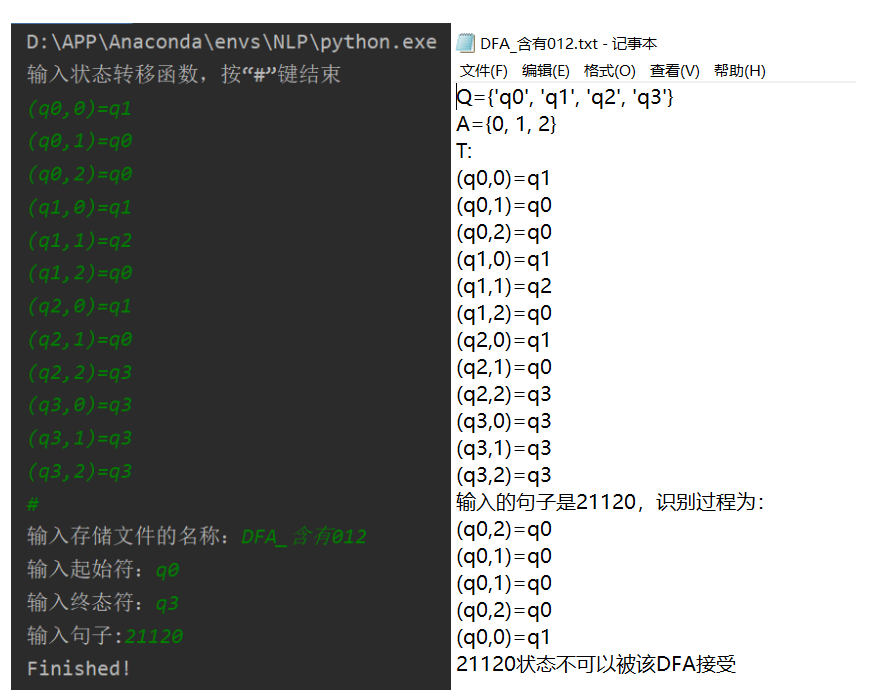


图5识别失败含012的句子

对于结果1，当q0读取1的时候，转移到状态q1，当q1读取状态0的时候转移到了状态q0，当q0读取状态0的时候转移到了状态q1，当q1读取状态0的时候转移到了状态q2，当q2读取状态0的时候转移到了状态q3，此时遍历结束，present指向了最后一个结点，并且该节点正好是终结符，自动机能够停止，因此101000是可以被该自动机所识别的。

对于结果2，当q0读取1的时候，转移到状态q0，当q0读取状态1的时候转移到了状态q1，当q1读取状态1的时候转移到了状态q0，当q0读取状态0的时候转移到了状态q1，当q1读取状态1的时候转移到了状态q0，当q0读取状态0的时候转移到了状态q1，当q1读取状态1的时候转移到了状态q0，当q0读取状态0的时候转移到了状态q1，当q1读取状态0的时候转移到了状态q2，此时遍历结束，present指向了最后一个结点，但是该节点不是终结符，自动机不·能够停止，因此101010100是不可以被该自动机所识别的。

对于结果3，当q0读取0的时候，转移到状态q1，当q1读取状态0的时候转移到了状态q1，当q1读取状态1的时候转移到了状态q2，当q2读取状态1的时候转移到了状态q0，当q0读取状态2的时候转移到了状态q0，当q0读取状态2的时候转移到了状态q0，当q0读取状态0的时候转移到了状态q1，当q1读取状态1的时候转移到了状态q2，当q2读取状态2的时候转移到了状态q3，当q3读取状态2的时候转移到了状态q3，当q3读取状态2的时候转移到了状态q3，当q3读取状态1的时候转移到了状态q3，当q3读取状态1的时候转移到了状态q3，当q3读取状态0的时候转移到了状态q3，当q3读取状态0的时候转移到了状态q3，此时遍历结束，present指向了最后一个结点，并且该节点是终结符，自动机能够停止，因此001122012221100是可以被该自动机所识别的。

对于结果4，当q0读取2的时候，转移到状态q0，当q0读取状态1的时候转移到了状态q0，当q0读取状态1的时候转移到了状态q0，当q0读取状态2的时候转移到了状态q0，当q0读取状态0的时候转移到了状态q1，此时遍历结束，present指向了最后一个结点，但是该节点不是终结符，自动机不·能够停止，因此21120是不可以被该自动机所识别的。

1. 实验总结

主要总结所遇到的问题及其解决方法（可以是成功解决的方法或部分解决方法或无法解决的原因）。

**问题1：**在将运算过程和自动机五元组成分写道自动机时，我发现结果出现如下图6所示的乱码。

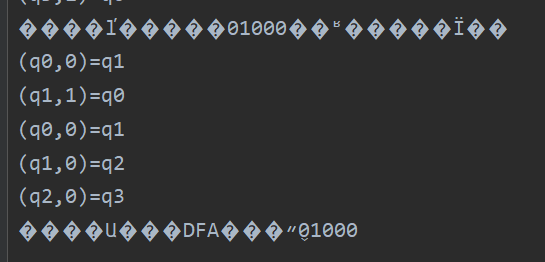


图6 出现乱码截图

**分析1：**造成这种的原因可能是因为电脑自身的文本解码器默认不是该编码对应的解码方式，因此会出现这种现象。但是解决方法一般不建议更换电脑的解码方式，因为很可能导致其他正常的文本文档出现乱码，解决这个问题的最好方法就是在写文件时规定他的编码格式为Utf-8格式，这样的话写进去的文件一直都是Utf-8格式的，就不会出现报错了。修改代码的位置及修改后的代码如下图7所示。

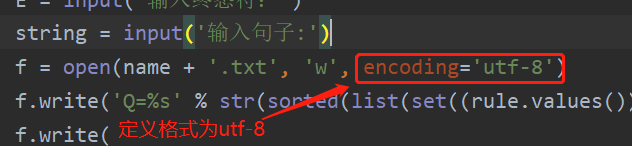


图7 乱码解决方案图