**深 圳 大 学 实 验 报 告**

**课程名称： 编译原理**

**实验项目名称： 词法分析程序设计**

**学院： 计算机与软件学院**

**专业： 软件工程**

**指导教师： 尹剑飞**

**报告人： 张欣杰 学号： 2020151091 班级： 软工02**

**实验时间： 2023年3月11号-2023年4月08号**

**实验报告提交时间： 2023/4/08**

**教务处制**

|  |
| --- |
| **实验目的与要求：**  **目的：**针对状态矩阵、NFA、DFA、正规式、确定化算法、最小化算法、Thompson算法等词法分析应用问题，设计并实现相应的解决方案，通过设计FA相关类族，以及实现前述几个重要的算法，既加深对抽象的词法、自动机、正规式等形式语言理论基础概念的理解与掌握，也加强对面向对象程序编写能力和计算思维的培养。  **要求:**  **第一部分： 无符号数的识别及DFA的应用**  这部分又分为2个小实验   1. 输入：一个文本文件（源代码文件）   输出：将源代码中的无符号数识别出来并输出到另一个文件中  示例：如果输入是“123\*abc+def/99.2+9.9E+c”，那么输出是：（数字, 123），（其它，\*abc+def/），（数字，99.2），（其它，+），（异常，9.9E+c）  说明：其它是非数字打头的字符串；异常是数字打头，但最后却是不符合定义的无符号数。   1. 假设：用字符“ABCDEFGHIJK”（大写）分别表示数字0..9和.、E、+、-，那么，字符串“BCLD”表示数字“12E3”=12000；   输入：一个文本文件  输出：将隐藏在文本文件中的有效无符号数识别出来。  示例：如果输入是“BCD\*abc+def/JJKC+JKJL+c”，那么输出是：（数字, 123）、（数字，99.2），无效（异常）的无符号数不输出  **第二部分：DFA/NFA的读写及确定化、最小化算法 （选做）**  这部分又分为3个小实验   1. 输入：一个文本文件（格式自定义）   输出：读入文件中的DFA/NFA，创建对应的DFA/NFA对象，再写回另一文件中。  示例：如果输入是“f(S,a)=A, f(A,b)=B, {B}” //假设默认开始状态是S，  那么输出是：K={S，A，B}；Σ={a,b}；f(S,a)=A, f(A,b)=B；S；Z={B}   1. 输入：上一实验输出的NFA文件   输出：将读入的NFA进行确定化后，输出结果，需要考虑有/无ε的情况。   1. 输入：上一实验输出的DFA文件   输出：将读入的DFA进行最小化后，输出结果。  **第三部分：正规式及其应用**  这部分又分为3个小实验   1. Thompson方法的实现   输入：一个正规式  处理：创建与正规式对应的NFA对象，再将NFA对象写回文件，格式与前述实验相同，使得能够读回；  示例：如果输入是“a\*b”，那么输出是：K={A , B , C , D , E , F}；Σ={a, b}；f(A,ε)=B, f(B,a)=C, f(C, ε)=B, f(C, ε)=D, f(A, ε)=D, f(D, ε)=E, f(E, b)=F；A；Z={F}  说明：优先级处理可用栈或递归，思路见算符优先分析法。概要来说，判断当前运算符与下一个运算符间的优先级，当前≥下一个则计算（产生NFA），否则入栈或递归。   1. 从正规式到DFA   将上述实验与前述实验连接起来，使得读入一个正规式文件，能输出一个DFA。正规式文件格式自定，通常每行一个正规式。  示例：如果正规式文件包括2行，每行分别是“a”和“b”，那么，输出的DFA实际上是“a | b”对应的DFA。   1. 字符串的识别   输入：一个正规式文件和一个字符串文件  输出：判断字符串文件中的每个字符串，能否被正规式对应的DFA所识别  其次，再给每个正规式增加一个类别，识别到给定字符串符合某个特定正规式时，输出该类别。  示例：如果输入  a\* 类A  b 类B  那么对字符串aaa输出：aaa，类A |
| **方法、步骤：**  要完成本实验，依据实验要求进行分解，需要从数据结构设计和面向对象设计角度，思考并回答下述问题：   1. 如何识别无符号数？   如何判断一个输入的字符串中，包括无符号数字？如何判断无符号数的开始？中间字符及结束？     1. 根据题意，可以根据无符号数的识别画出如上状态转换图，首先输入一个字符串，如果该字符串为无符号数，则该串的首字符必为数字，若不为数字，则为不合法的终结状态，即该串不是无符号数。 2. 当第一个字符为数字，则进入状态1，继续接收输入的字符，当字符持续为数字时则保持在状态1，若字符为“.”或者“E/e”时，则跳转到状态2或者4。若输入结束，1为合法终结态，则输入的字符串为合法的无符号数。 3. 当第一个字符为“.”则跳转到状态3，若接下来的字符为数字则跳转到状态2，否则为非法状态，输出非无符号数。 4. 在状态2时接收字符，若字符为数字则保持为状态2，若输入结束，2为合法终结态，则输入的字符串为合法的无符号数。若输入为“E”则跳转到状态4。 5. 进入状态4时，若接收的字符为“+/-”则跳转到状态5，否则为非法终结态，即字符串为非无符号数。若接收的字符为数字，则跳转到状态6。 6. 状态5接受字符，若字符为数字，则进入状态6，否则为非法状态。 7. 状态6只接收数字字符，否则为非法状态。 8. 合法终结态为状态1、2、6，在其他状态结束输入均为非法终结态。 9. 如何实现简单的字符映射加解密算法？   一种简单的字符映射加解密算法是将每个字符映射到另一个字符上，可以使用一个固定的映射表来进行加解密操作。可以自己创建一个映射表对字符进行映射加密，或者最简单的方法时将字符转换为ascii，然后使用ascii计算与某个固定字符的差值对字符进行加密。  解密时只需要按照解密时所使用的映射规则对字符逐一进行反映射即可对加密字符的解密操作。  在本次实验用，可以直接选用一个映射表，将字符一一映射，解密进行映射解密即可。   1. 如何定义DFA/NFA对象，特别的，对自动机里面的映射函数f如何定义、存储？（注意这里的“对象”，指用面向对象的程序设计思维方式（内部数据结构、消息）来定义一个DFA类、一个NFA类。）   在面向对象的程序设计思维方式下，可以定义一个DFA类和一个NFA类来表示有限状态自动机。这两个类都应该包含有限状态自动机的必要属性和方法，包括状态集、输入字母表、起始状态、接受状态、转移函数等。  对于转移函数f，可以使用一个字典来存储，字典的键为元组(state, input)，表示从状态state接收到输入input时的转移，字典的值为转移后的状态。这种方式可以有效地存储和查找转移函数，因为它能够充分利用Python中字典的查找性能。   1. 如何实现确定化算法？（选做）   确定化算法需要把一个旧的状态集合作为一个新的状态，这种情况如何处理？又如何求某个状态的ε闭包？  请在这里，补充完善你的设计思路，完成后，请删去这句话。   1. 如何实现最小化算法？（选做）   最小化算法需要对原状态集进行等价类划分，这个划分要如何处理？  请在这里，补充完善你的设计思路，完成后，请删去这句话。   1. 汤普森方法需要处理算符的优先级，该如何处理？如何进行递归调用？   当处理到运算符时，递归调用Thompson方法，将左右两个子表达式转换成对应的NFA。对于星号、竖线或点号运算符，递归调用Thompson方法的方式略有不同：   1. 对于星号运算符，先递归处理星号运算符右边的子表达式，然后将结果存储在一个临时NFA对象中。 2. 接下来，创建两个新状态，一个新的起始状态和一个新的接受状态。 3. 修改临时NFA对象，将其可以接受0个或多个字符，并且将其接受状态指向新的接受状态。 4. 将新的起始状态指向临时NFA对象的起始状态，并将新的接受状态作为返回值。 5. 对于竖线运算符，先递归处理竖线运算符左边和右边的子表达式，然后将结果存储在两个临时NFA对象中。 6. 接下来，创建两个新状态，一个新的起始状态和一个新的接受状态。 7. 修改两个临时NFA对象，将其可以并行接受输入字符串，并且将其起始状态分别指向新的起始状态。 8. 将新的接受状态指向两个临时NFA对象的接受状态，并将新的起始状态和新的接受状态作为返回值。 9. 对于点号运算符，先递归处理点号运算符左边的子表达式，然后递归处理点号运算符右边的子表达式，并将结果存储在两个临时NFA对象中。 10. 修改两个临时NFA对象，将左边NFA的接受状态指向右边NFA的起始状态。 11. 将左边NFA的起始状态和右边NFA的接受状态作为返回值。   在递归调用中，将Thompson方法的返回值作为参数传递给上层的Thompson方法。通过这种方式，可以逐层地构建NFA，最终得到一个完整的NFA。   1. 如何根据给定DFA，判断并输出给定输入串的类别？   给定一个DFA和一个输入字符串，我们可以使用该DFA来判断输入字符串是否属于该DFA所表示的语言，并输出相应的类别。具体步骤如下：   1. 首先，将输入字符串的第一个字符作为当前状态的输入字符。 2. 然后，查找DFA中是否存在从当前状态出发，输入字符为当前输入字符的转移。如果存在，则将当前状态更新为该转移的目标状态，并继续处理输入字符串的下一个字符。如果不存在，则说明该输入字符串不属于该DFA所表示的语言，可以直接输出不合法的信息。 3. 当输入字符串中的所有字符都被处理完毕时，判断当前状态是否为该DFA的终止状态。如果是，则说明该输入字符串属于该DFA所表示的语言，可以输出相应的类别；否则，说明该输入字符串不属于该DFA所表示的语言，可以直接输出不合法的信息。   注意，对于一个DFA，可能存在多个终止状态，因此需要根据实际情况对当前状态进行判断。如果当前状态属于该DFA的终止状态集合，则说明该输入字符串属于该DFA所表示的语言。 |
| **实验过程及内容：**   1. 第一部分 2. 在该部分中，首先定义一个状态转移矩阵，根据上述状态图画出以下状态转移矩阵，其中如果在状态转移过程中，跳转到other的时前一个状态为1、2、6为合法终结态，否则为非法终结态。  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 状态  输入 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | digit | 1 | 1 | 2 | 2 | 6 | 6 | 6 | / | | dot | 3 | 2 | / | 7 | / | / | / | / | | E/e | / | 4 | 4 | / | / | / | / | / | | +/- | / | / | / | / | 5 | / | / | / | | other | 7 | 7 | 7 | 7 | 7 | 7 | 7 | / |   在python可以定义一个有向图transition来表示状态转移；在有向图中，每个节点表示一个状态，每条边表示一个输入字符，边的权重表示从一个状态转移到另一个状态所对应的输出类型。在该有向图中，每个节点的出边的字典表示从该节点转移到其他节点所需的输入字符和转移后的状态。  部分状态转移函数get\_state()代码如下：    传入参数为initial和receive两个参数，返回为transition+.get(initial).get(receive)，也就是起始态接受某一字符后的下一个状态。  在主函数中，首先使用python的docx模块打开doc文件，然后将文件中的内容读取到字符串中，然后使用split方法按照空格分隔的方式转换为列表，然后遍历列表，将列表中的空串给剔除。    将列表传入子函数is\_unsigned\_integer函数中，在该函数中遍历列表中的元素，首先判断元素中是否有数字，若没有数字，则直接将“（其他，s）”写入文本文件中；如果有数字则将该字符串传入子函数get\_digit中进一步将字符串中的无符号数提取出来，将提取出来的各个类型的串写入到创建的result文本文件中    在is\_digit子函数中，首先将状态设置为’start’，遍历字符串，调用get\_state方法获取当前状态，然后根据状态进行下一步操作，在这里需要注意的是状态为’end’的几个状态，在遇到end状态时则将结果写入到列表中去，然后重新设置开始状态进行下一次遍历。    需要注意的时，在遍历完一个字符串后，还需要对字符串缓冲区buffer进行检测，buffer如果不为空，则按照当前状态将结果写入列表  中。     1. 在part1-1的基础上增加一个映射表     然后在遍历列表的时候，不进行数字判断改为加密判断    在子函数中遍历加密字符串时，也就是遍历含有数字的字符串时，使用映射表进行逐一解密，然后按照原本步骤进行输出即可。    完成解密过程之后，提取无符号数过程与第一题时相同的，只不过只需要将数字输出，非数字 部分不需要输出。   1. 第三部分 2. 首先定义一个state\_node类，使用该类来表示状态的转移      1. 定义一个regex类，用来存储正则表达式，然后在该类里边定义一个函数，将正则表达式转换为后缀表达式      1. 定义一个nfa类，对需要用到的状态码，字母表，状态转换表等一一初始化     首先在主函数中，使用输入函数接收正则表达式，然后调用regex类的转换函数将正则表达式转换为后缀表达式，然后初始化NFA类对象，调用state\_init函数对初始状态进行初始化    然后对后缀表达式进行操作数栈和操作符栈，初始化完成后，进行NFA转换。    针对字母、操作符一一进行运算，我在这里对闭包运算进行了判断，如果与或运算时一起的话则进行或闭包运算。在这里为了防止二义性，我使用了·也就是点乘符号，用来连接，也就是对表达式进行分割。  在完成运算过后，使用调用print\_nfa函数对nfa进行输出 |
| **实验结论：**   1. 实验1 第一部分   输入为question.docx，也就是实验材料所给的问题，输出为result文件，输出结果如下：    可以看到，能够正确输出文件中的无符号数   1. 第二部分   首先对question.docx进行加密，加密完成后的文本如下    然后对文本文件进行解密，再提取无符号数如下：     1. 实验输入a\*·b，输出如下     输入1\*·0·(0|1)\*,输出如下： |
| **心得体会：**  通过本次实验，我对词法分析程序的实现原理和实践方法有了更深入的了解。我还学会了使用Python中的re模块进行正则表达式匹配，同时还极大地提升了我的python编程的能力。  本次实验难度对我来说较大，遇到的困难较多，但是通过查询资料和学习之后，在解决问题的同时也收获了许多。在本次编写的代码中，可能会些许错误，在后续的学习中我会继续努力，努力完成后续实验及练习。 |
| 指导教师批阅意见：  成绩评定：  指导教师签字：尹剑飞  年 月 日 |
| 备注： |