Lex 实验报告

a)Motivation/Aim

完成将输入的RE表达式通过JAVA程序逐步转化，1.规范RE表达式，2.转为后缀，3.RE后缀转NFA，4.NFA转DFA，5.DFA转DFA0，这些过程全部实现，并可识别多种RE表达式，并可以通过读取输入的input.txt 正确判断每一个单词所对应的正确的token。返回token序列以及对应的yylval。

b)Content description

src 下面分为三个包，exception 表示异常，process是过程中各个步骤的中间产生式，由RE到DFA0，utility是各种用到的数据结构以及定义一些全局变量和re的优先级。

输入的REs.l文件是token对应的RE表达式，input.txt为输入的需要解析的文件。

c)Ideas/Methods

1.先将RE中的各个课转换的表达式进行格式化，如[a-z]变为（a|b|\*\*\*\*\*|z），a{2,4}变为（aa|aaa|aaaa）等等，并加上连接符方便后缀表达式的转化。实现了各种RE表达式。

2.转化为RE后缀表达式

3.利用TOMPSON算法定义一些基本的RE 如 遇到字符 遇到？+ \* | ▪ 等初始化如何定义NFA，通过逐个读RE后缀表达式，进行NFA表达式的压栈和出栈生成最终的NFA队列，每一个RE对应一个NFA，TOKEN也同时读入NFA记录。并定义NFA的初始和终止状态

4.将NFA转化为DFA，我采用的是不合并NFA的办法，每一个NFA单独处理，通过求ε闭包和每条边对应的下一个状态，循环生成下一轮还未存在过得状态并记录在DFA边的集合中，生成可查询的DFA。若是DFA边的尾结点包括了NFA中的终止状态，则DFA边尾结点对应的状态设置为终止状态。

5.分组为终止状态和非终止状态，一旦两个状态每个对应的相同边的尾状态不在一个组内，则将其分为两个组，然后循环直到分组完全，每一组选一个状态代替其他状态，并生成DFA0。

6.读入input.txt文件逐行读取，遍历每一个DFA0。每一行逐个字符读取如DFA0，如果匹配，则读下一个，否则查看上一个终止状态匹配点，记录此时的指针的位置对应的token序列，以及长度和对应的文本。若是之后有其他DFA0匹配，比较指针对应的长度和优先级来确定所对应字段的TOKEN序列为多少，遍历完Token序列后将头指针指向TOKEN序列对应的位置，并进行下一字符的查询。知道文件所有完毕。如果每一个DFA0都不能产生TOKEN，则报错。

d) assumption

REs.l中每一行的格式为 “TOKEN/yylval RE”

即每一行必须包括token 空格 re表达式。如果有/则看是否后面有yylval，如果有的话就直接保存 如 RELOP/EQ == ，如果没有yylval的话，就记录此token，之后从文本文件中读取并记录真实的yylval，如 “NUMBER/ [0-9]+(\.[0-9]+)?(E[\+\-]?[0-9]+)?”

e）Thompson算法，具体部分见前面的实验报告

重要的数据结构

Token\_RE类中包含token yylval 以及对应的RE表达式 都是字符串，用来读取RES.l 并进行初步的分析

Token\_NFA类中包含token yylval 以及对应的NFA结构 用来分析RE后缀并保存NFA结构，NFA结构中包括了每一条NFA边（起点序号，边的char对应的值，终点序号）的list

Token\_DFA类中包含token yylval 以及对应的DFA结构 用来分析NFA并保存DFA结构，DFA结构中包括了每一条DFA边（起点状态，边的char对应的值，终点状态，【每一个状态包括其对应的ID和NFA的序号方便之后进行优化操作】），之后直接进行优化（DFA0和DFA的数据结构相同）

g）核心算法之前已经讲清楚

h）例子

REs.l

THEN then  
ELSE else  
PUBLIC public  
STATIC static  
SSS (a▪b)\*c+

myRE r{2,3}(a|b){2,3}  
ε [ \t\n]+  
IF if  
MAIN main  
ID/ [a-z]([A-Za-z0-9])\*  
NUMBER/ [0-9]+(\.[0-9]+)?(E[\+\-]?[0-9]+)?  
RELOP/LT <  
RELOP/GT >  
RELOP/GE >=  
RELOP/NE !=  
RELOP/LE <=  
RELOP/EQ ==  
RELOP/E =

Input.txt

ab main = abab123  
 10.98 10E-3  
ababc  
  
 >=

output.txt

ID MAIN RELOP ID   
 NUMBER NUMBER   
 SSS   
   
 RELOP

说明，可以正确辨析ID和别的定义的保留字，如第一行

可以读取10.98 以及带E的浮点数是NUMBER 的token序列 ，如第二行

可以读到SSS是自己定义的表达式而不是优先级比他低的ID，如第三行

可以读取带yylval的re并保留yylval显示token，如第四行

可以自己随便添加这种程序允许的RE进行分析。

i）Problems occurred and related solutions

判断是哪个DFA TOKEN序列的问题前面已经说明。

如果在RE转换时出现问题，如括号不对应，{ } 里面有别的格式错误等等，会直接抛出错误停止程序继续进行，，并抛出第几行是错误的表达式。

在RE到NFA转换的时候，如出现栈中不是只有一个NFA，则抛出异常，终止程序，提醒哪个RE表达式不符合规范。

读input.txt的时候，如果所有的DFA0都不能解析出是哪个Token，则进行报错。让其检查输入文件是否符合定义的REs.l

j）Your feelings and comments

已经努力完成自己想完成的所有任务。