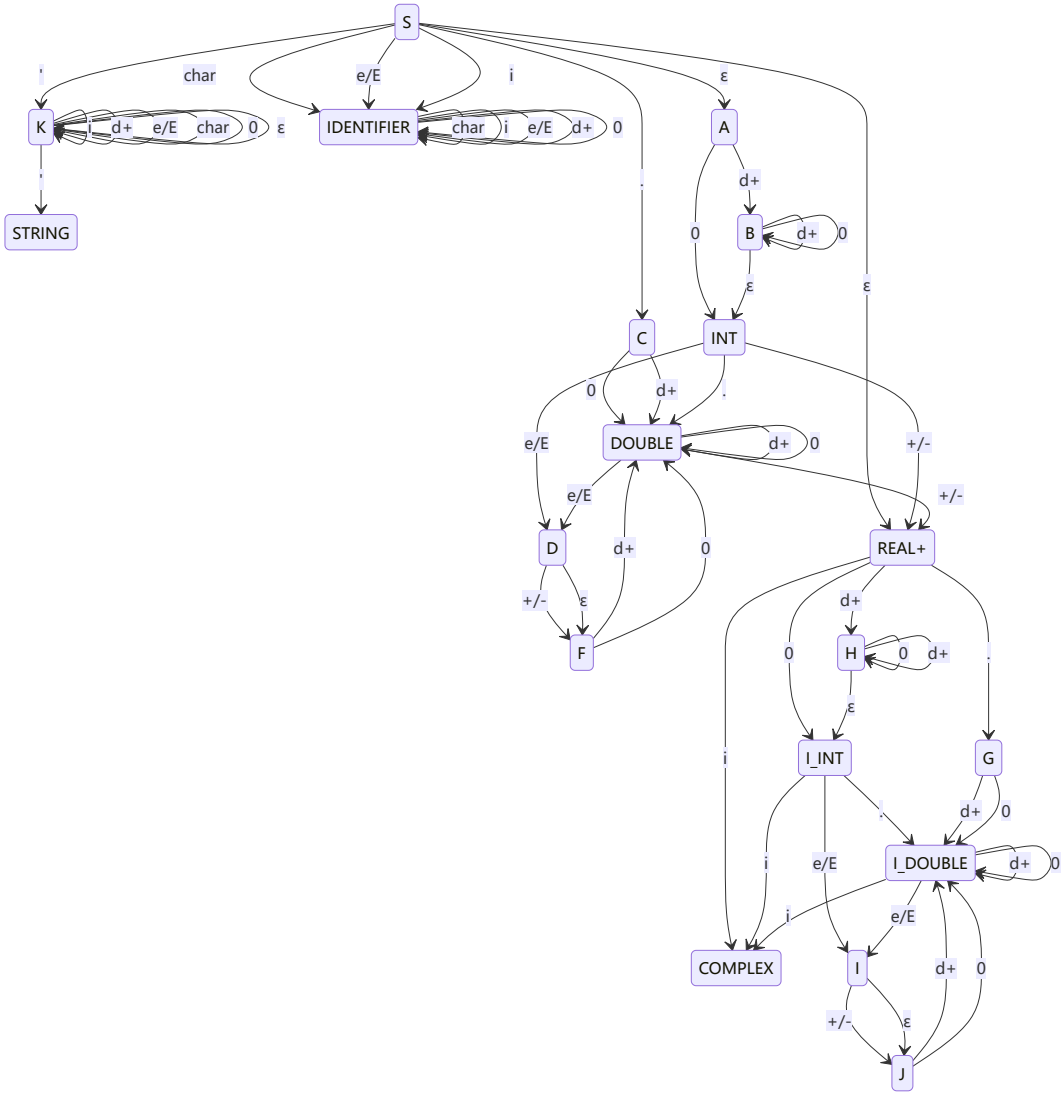
任务一 词法分析

1 设计思路

1.1 词法设计

1.1.1 NFA设计

在词法分析程序中，为分析器设计的文法NFA如图所示：



其中可接受状态为`IDENTIFIER`，`STRING`，`DOUBLE`，`COMPLEX`，分别为标识符，字符串，浮点数，复数

转换边解释：

`i`：字母i

`e/E`：小写字母e或大写字母E

`char`：其他字母字符

`d+`：1-9的数字字符

`0`：字符0

`+/-`：字符+或字符-

`ε`：空输入

在处理前先对所有字符进行分类，简化了NFA的复杂程度

1.1.2 关键词，操作符，限定符设置

关键词：`for`, `while`, `if`, `else`, `return`, `break`, `continue`, `def`, `class`, `int`, `double`

赋值操作符：`+=`, `-=`, `*=`, `/=`, `*=`, `//=`, `%=`, `=`

二元运算符：`and`, `or`, `xor`, `==`, `**`, `+`, `-`, `*`, `/`, `//`, `%`, `in`, `<`, `>`, `<=`, `>=`

单目运算符：`!`, `not`

限定符：`.` `:` `,` `;` `[` `]` `{` `}` `(` `)`

1.2 数据结构的设计

本程序的NFA使用python的字典，是一种索引数据结构.

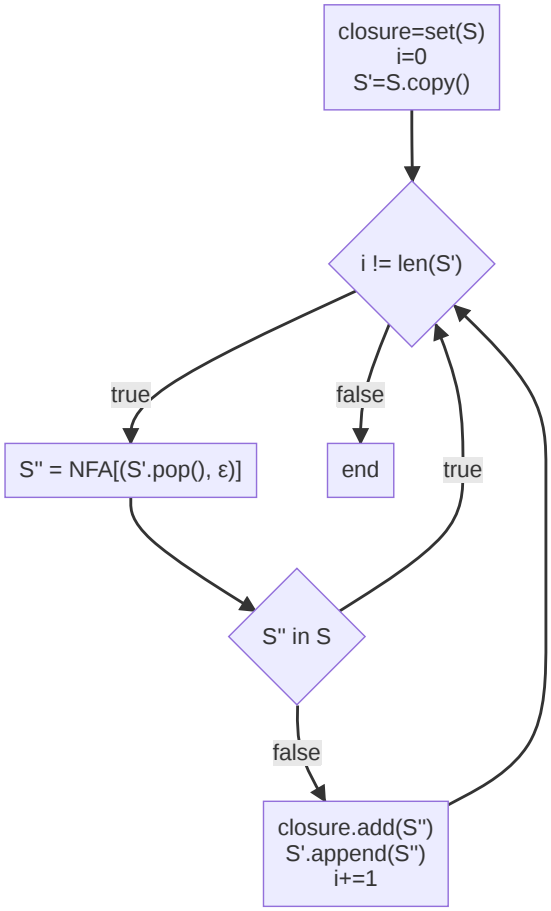
设一条正规产生式为$f(A,t)=Z$

具体格式为NFA[(A, t)] = Z. A, t作为键，Z作为值存储，方便查找，降低算法复杂度。由于NFA中$f(A,t)$可能有不止一种的状态，Z用set类保存所有f(A,t)的状态

而DFA中f(A, t)只有一种状态，因此Z不再用set保存，直接赋值。

1.3 ε-closure的计算

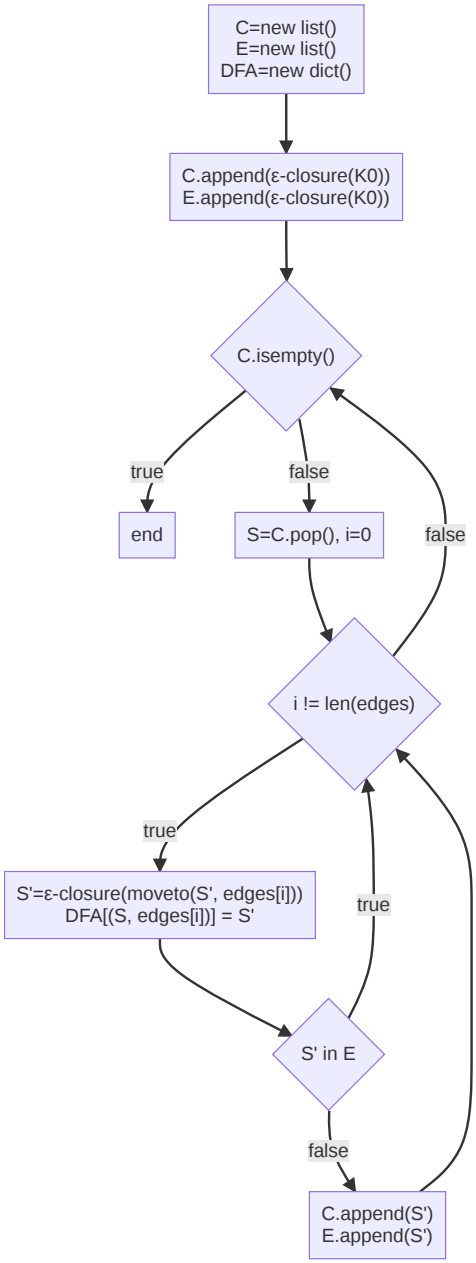
算法流程图如下：



遍历集合中的每一个元素，查找对应的ε边转换，加入到集合中，循环操作，直到集合不再增大为止

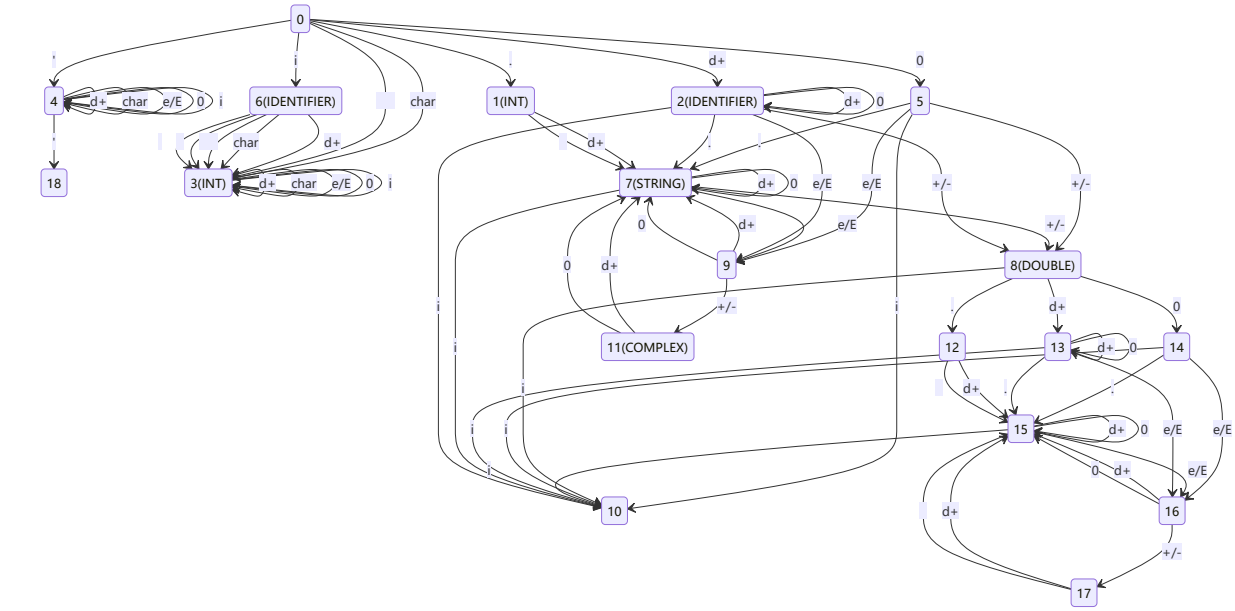
1.4 NFA到DFA转换

算法流程图如下：



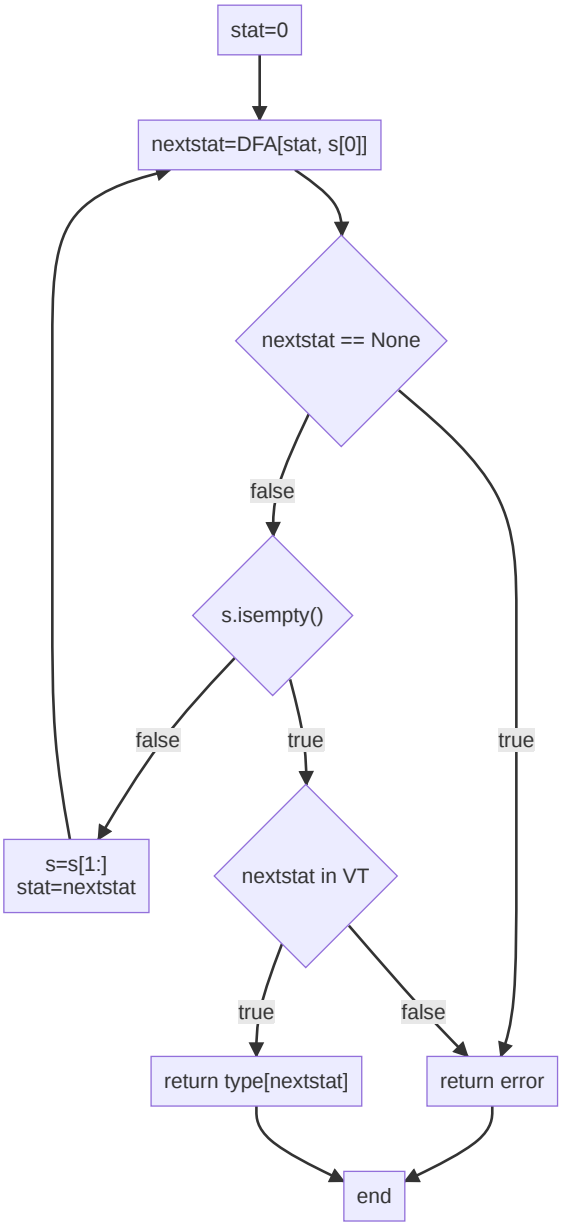
NFA到DFA转换采用了图遍历DFS算法，在这种情况下，图有可能是有环的。因此除了需要用一个队列C维护未被访问的状态，还要用一个队列维护已经访问过的数组E，用来防止回边导致的死循环。

NFA经过转换后的DFA如图所示：



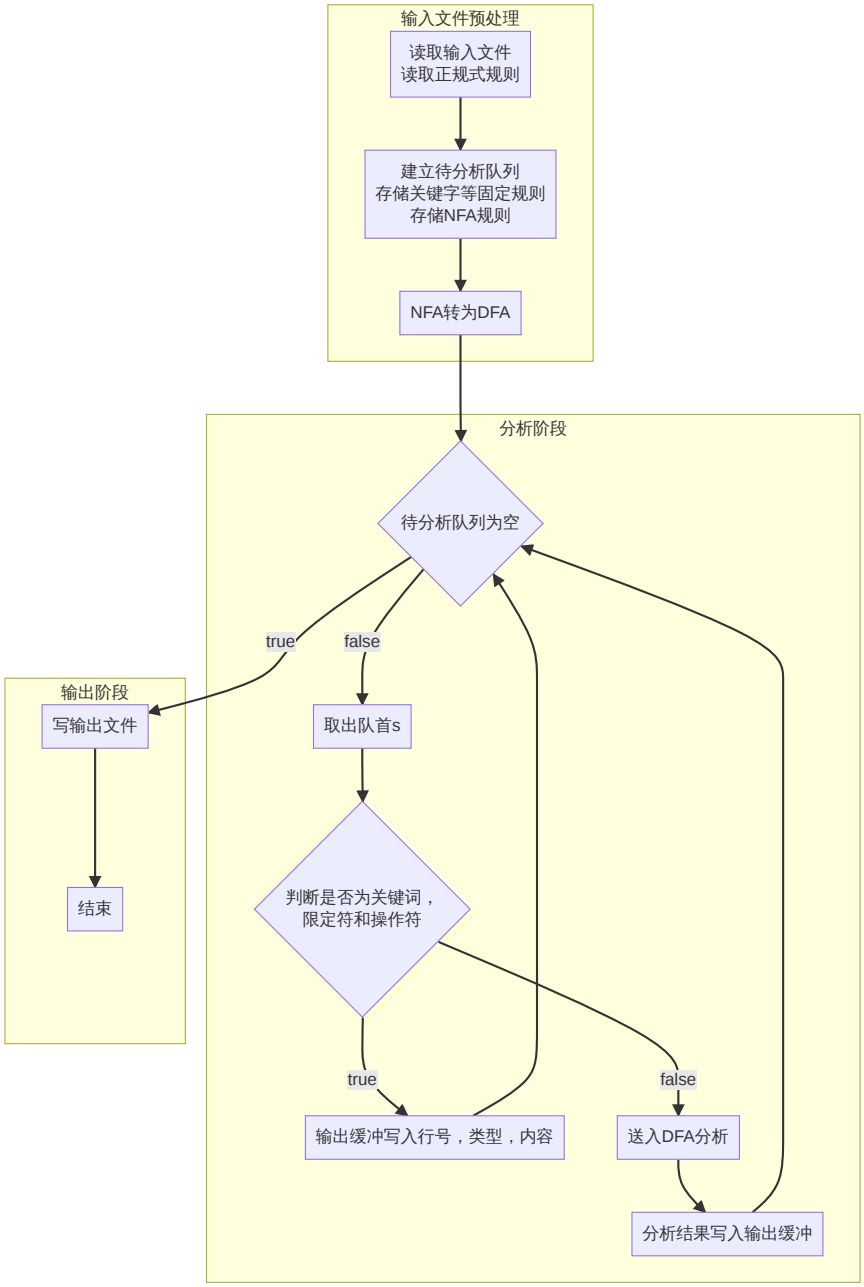
1.5 DFA和输入字符串的匹配

算法流程图如下



DFA的可接受状态为串中所有字符处理完毕，并且最终状态必须为可接受状态。其他情况（1. 查询不到转换状态 2. 最终状态不为可接受状态）均返回error

1.6 总流程图



2 代码函数解释

`def typeof(ch)`: 对字符进行分类，分类为d+,0,char,e/E,i类型（详见2.1.1节）
`def closure(NFA, depts)` 计算ε-closure，NFA：NFA语法，depts原始集合
`def NFA2DFA(NFA)` 计算NFA转换的DFA
`def Parse(DFA, s)` 通过NFA判断待分析串的类型

3 程序说明

词法分析的程序的可执行文件名称为`lex_parser.exe`，源代码文件为`lex_parser.py`

3.1 操作说明

在命令行输入命令 ./lex_parser -h获取帮助信息：

```
PS C:\Users\Anderson\Desktop\Gitworkspace\Compiler> ./lex_parser -h
usage:
lex parser by yuangyan
Courseworkwork for semester2, 2022
 [-h] [-s] [-i] [-r] [-o]

optional arguments:
  -h, --help      show this help message and exit
  -s, --showDFA   show DFA coverted from input NFA
  -i , --input    input file name, 'lex.txt' by default
  -r , --rule     rule file name, 'lex_rule.txt' by default
  -o , --output   output file name, '<input file name>_parsed.txt' by default
```

在词法分析程序中共有5个可选参数，分别为：

-h, --help: 显示帮助信息
-s, --showDFA: 在程序运行的过程中打印从输入NFA转换成的DFA
-i, --input: 选择输入待分析文本路径，不选该参数则默认为同目录下'lex.txt'文件
-r, --rule: 选择词法分析规则路径，不选该参数则默认为同目录下'lex_rule.txt'文件
-o, --output: 选择输出token表路径，不选该参数则默认为同目录下'<输入文件名>_parsed.txt'文件

输入命令格式为
./lex_parser (-i [待分析文本路径]) (-r [词法分析规则文件]) (-o [输出token路径]) (–showDFA)
例如：./lex_parser -i input.txt -o output.txt -r rule.txt --showDFA

3.2 输入输出样例

输入文件lex.txt:

> identifier123
> 123identifier
> $identifier
> 0.12
> .12
> 12
> 12.
> 12.00
> 1.2e+3
> 3i
> 12+3i
> 1.2+3.4e+5i
> 'YuangYan'
> 'YuangYan
> while(a<b){
> c[d] = e.getX()
> }

输出的token list，路径为lex_parsed.txt

```
1 IDENTIFIER identifier123
2 invalid_syntax 123identifier
3 invalid_syntax $identifier
4 DOUBLE 0.12
5 DOUBLE .12
6 INT 12
7 DOUBLE 12.
8 DOUBLE 12.00
9 DOUBLE 1.2e+3
10 COMPLEX 3i
11 COMPLEX 12+3i
12 COMPLEX 1.2+3.4e+5i
13 STRING 'YuangYan'
14 invalid_syntax 'YuangYan
15 KEYWORD while
15 DELIMITER (
15 IDENTIFIER a
15 BINARY_OPERATOR <
15 IDENTIFIER b
15 DELIMITER )
15 DELIMITER {
16 IDENTIFIER c
16 DELIMITER [
16 IDENTIFIER d
16 DELIMITER ]
16 ASSIGNMENT_OPERATOR =
16 IDENTIFIER e
16 DELIMITER .
16 IDENTIFIER getX
16 DELIMITER (
16 DELIMITER )
17 DELIMITER }
```

3.3 输入样例解释

在lex_parsed.txt文件中每行格式为[行号] [类型] [内容]
在设计的文法规则中
1） 标识符只能由字母开头，因此123identifier和$identifier被识别为invalid_syntax
2） 0.12，12.，12.00 12e+3均能识别为`double`类型
3） 复数的实部和虚部均可为`int`和`double`类型，兼容科学计数法，如1.2+3.4e+5i
4） 字符串常量由成对的单引号标识，'YuangYan缺少一个单引号，因此被识别为`nvalid_syntax`
5） 可以识别连在一起的不同类型，如while(a<b)被识别为关键字，限定符，标识符，赋值操作符，限定符

当加入`–showDFA`参数时，程序运行过程中会打印转换后的DFA，如下：

```
PS C:\Users\Anderson\Desktop\Gitworkspace\Compiler> ./lex_parser --showDFA
DFA:
0--d+-->1   (INT)
0--.-->2
```

```
0--e/E-->3   (IDENTIFIER)
0--i-->4   (COMPLEX)   (IDENTIFIER)
0--0-->5   (INT)
0--'-->6
0--char-->3   (IDENTIFIER)
6--d+-->6
6--e/E-->6
6--i-->6
6--0-->6
...
```

3.4 输入规则解释

在输入规则中，每行的第一列标识规则类型，分别为NFA，Keyword，AssignmentOperator，UnaryOperator，Delimiter和DFAType

1） 规则为`NFA`：

设NFA产生式为$f(A,t)=Z$，A在第二列，t在第三列，Z从第四列开始

2） 规则为`Keyword` `AssignmentOperator` `UnaryOperator` `Delimiter` `DFAType`均在第二列添加指定的字符串

3）规则为`DFAType`

用于表示DFA的可接收状态，在第二列添加指定的字符串，下图为部分lex_rule.txt内容

> NFA IDENTIFIER char IDENTIFIER
> Keyword while
> AssignmentOperator +=
> BinaryOperator and
> UnaryOperator !
> Delimiter .
> DFAType INT