

编译原理实验设计说明



实验一 RE转NFA

学生姓名 杨振华

学生学号 201626811225

学生班级 软件工程 1603

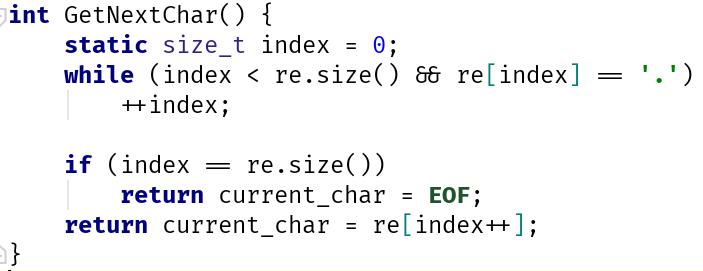
任课教师 张端

提交日期 2018年10月22日

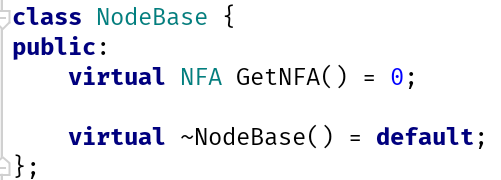
**计算机科学与技术学院**

1. 备注
2. 本实验代码采用C++17特有的特性，低版本编译器无法通过编译。
3. 除了output.txt，本程序还会生成output.gv，可用Graphviz软件画出NFA，方便查看NFA。
4. 本实验中所有出现的ε均由e代替。
5. 设计说明

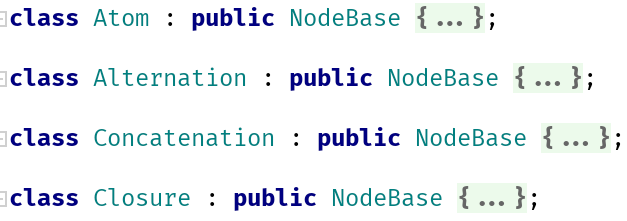
经过分析，要将RE转为NFA首先需要建立语法树，然后根据语法树逐层递归生成NFA。在这之前，首先需要将RE分解为字符流，分解过程中直接忽略了“.”，局部代码如下。



接下来需要构建语法树，先考虑语法树的节点。由于进行不同操作的节点类型不同但都又输入同一“节点”类型，这里采用类的继承实现。首先，基类如下。



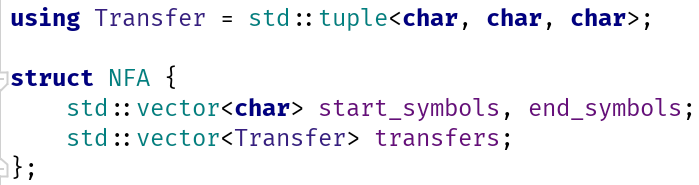
其中GetNFA函数用于后续递归产生NFA。接下来，每一操作都继承基类，形成一个子类。



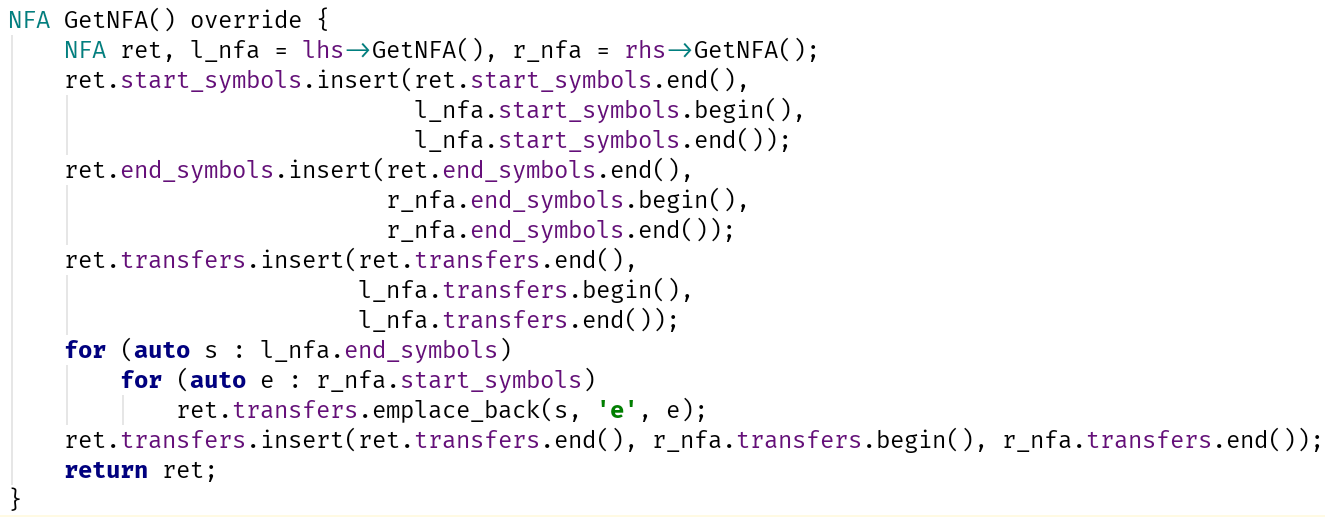
接下来就是进行语法分析，这里采用递归的方式实现。例如，在最顶层的分析过程中遇到“(”，进行递归，直至递归的函数内遇到“)”，之后递归函数返回括号内的整个语法树，交给顶层继续处理。由于代码比较长，这里不附加截图，具体请看ParseRE函数的实现。

最后，ParseRE函数返回一个指向整棵语法树的指针（处于内存管理的便捷，这里用了智能指针std::unique\_ptr）。对该树调用GetNFA函数即可递归获得最终的NFA。

接着看一下NFA结构的实现。



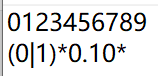
其中分别保存了初态、终态和转移函数。下面，我们来看一下每个节点子类继承基类并实现的函数GetNFA。以Concatenation类为例，主要是（根据其子节点）填充了NFA类的各个字段，然后将其返回。



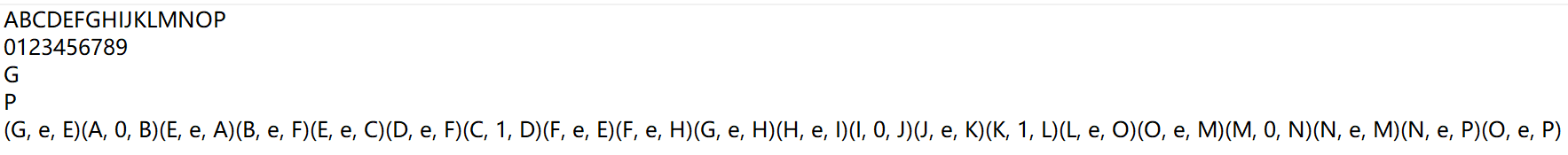
最后，在main函数里调用ParseRE和GetNFA函数，再将获得的NFA对象的所需字段打印出类即可。

1. 运行结果

对于题中给定的RE：



其output.txt为：



将生成的output.gv画图后：

