设计文档

DFA 算法 1

要求：　用 Java 编写，保证可扩展性

　　　　判断一个自动机是不是 DFA？是不是 NFA？

　　　　实现自动机的交并补操作。

　　　　实现自动机最小化的算法。

实验一：判断一个自动机是不是 DFA？是不是 NFA

通过遍历每一个状态的，这个状态的每一个变迁比较，在字符集合中 Σ 中每一个字符的状态变迁进行判断，若每一个状态的所有字符集只有一个变迁且没有 ε，则标记为 DFA，否则为 NFA。算法运行时间 $O(|Q||\Sigma|)$。

实验二：实现自动机的交并补操作。

对于补操作：$A=(Q,\sum,\delta,s,F)$ 是一个 DFA，那么由补操作产生的新 DFA 定义为：

$\overline{A}=(Q,\sum,\delta,s,Q-F)$。只要将 A 中的接受的状态设为不接受状态，同时把不接受的状态设为接受的状态就得到 $\overline{A}$。补运算的复杂度是 $O(|Q|)$。

对于交运算和并运算，有两个 DFA，$A_1=(Q_1,\sum,\delta_1,s_1,F_1)$ 和 $A_2=(Q_2,\sum,\delta_2,s_2,F_2)$，那么有这两个 DFA 创造出来的新的自动机定义为：$B=(Q_1\times Q_2,\Sigma,\delta_B,(s_1,s_2),M)$。其中

$M\subseteq Q_1\times Q_2$，(s_1,s_2) 为 B 的开始状态，δ_B 为 B 的转移函数，且做作如下定义：

$\forall q_1\in Q_1,q_2\in Q_2,\sigma\in\sum:\delta_B((q_1,q_2),\sigma)=(\delta_1(q_1,\sigma),\delta_2(q_2,\sigma))$

1、当 $M=F_1\times F_2$ 时，有上述方法得到的 B 就是 DFA A_1,A_2 的交运算。

2、当 $M=Q_1\times F_2\cup F_1\times Q_2$ 时，由上述方法得到 B 就是 DFA A_1,A_2 的并运算

交运算和并运算的复杂度都是 $O(|Q_1||Q_2||\Sigma|)$

实验三：实现自动机的最小化操作。

采用的方法是填表算法，基于如下递归的标记可区别的状态偶对的过程

-基础

　　若 p 为终态，q 为非终态，则 p 和 q 标记为可区别的。

-归纳

　　设 p 和 q 已标记为可区别的。若状态 r 和 s 通过某个输入符号 a 可分别转移到 p 和 q，

即 $\delta(r,a)=p,\delta(s,a)=q$，则 r 和 s 也标记为可区别的。

直到没有状态再改变。然后合并等价集合，获得最小化的自动机。设计复杂度为 $O(|Q|^2|\Sigma|)$。

设计类图如下：