正则表达式--自动机的转换工具

项目设计文档

Version1.0

作者：刘培尊 2009212542

欧阳明 2009212622

目录

1 项目说明

1.1 开发环境

操作系统：Microsoft Windows XP Professional 5.1.2600 Service Pack 3 Build 2600
机器配置：Memory：2G；
CPU：x86 Family 6 Model 23 Stepping 10 Genuine Intel ~2599 Mhz
实验环境：Eclipse-jee-galileo-win32

1.2 项目要求

项目的要求如下：

- 用Java编写，要求有良好的可扩展性
- 正则表达式转换为自动机
- 自动机转换到正则表达式

2 需求分析

2.1 总体分析

结合第 1 章中的项目要求，我们来分析项目的需求，根据要求 1，我们选择了 Eclipse 的开发平台。

在本项目中需要把读者的输入一个字符串转换为 DFA。在这之前需要判断读者输入的字符串是否为正则表达式，而这已经由正则表达式一题完成了相应的功能了。我们需要做的就是把一个确定为正则表达式的字符串转换为 DFA。

另外项目需要把一个 DFA 转换为一个正则表达式。

下图表示的是正则表达式、DFA 与 NFA 的关系：

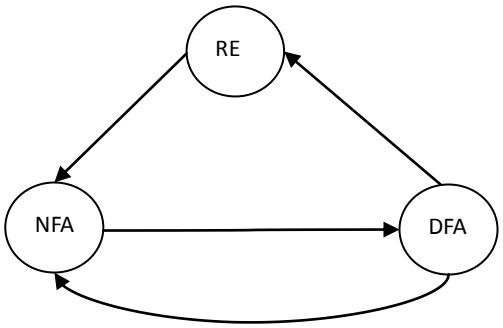


图 1 正则表达式、DFA 与 NFA 之间关系图

2.2 正则表达式到 DFA

要把正则表达式转化为 DFA，可以编写相关算法进行直接的转换，但毫无疑问，这将会增加程序设计过程的复杂性，并且代码的编写也会变得复杂，也会影响代码的优雅，降低代码的可阅读性。为此，我们把整个项目拆分为下面三个模块：

- 由正则表达式构造相应的 NFA；
- 把 NFA 转化为与其等价的 DFA；
- 最小化 DFA。

2.3 自动机到正则表达式的转化

这个转化，相对要简单些，我们只需要读取输入的自动机（DFA 或者 NFA）然后，然后把其转换为正则表达式即可。

3 概要设计

3.1 正则表达式到 NFA

这是本项目的一个重点，在把正则表达式转化为 NFA 的过程中，一般是使用 Thompson 构造法。

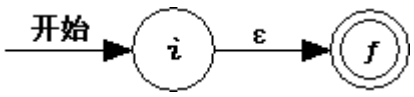
输入：字母表 Σ 上的正则表达式

输出：能够接受 L(r)的 NFA N

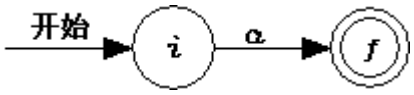
方法：首先将构成 r 的各个元素分解，对于每一个元素，按下规则 1 和规则 2 生成 NFA。

注意：如果 r 中记号 a 出现了多次，那么对于 a 的每次出现都需要生成一个单独的 NFA，之后依照正规表达式 r 的文法规则，将生成的 NFA 按照下述规则 3 组合在一起。

- **规则 1** 对于空记号 ε，生成下面的 NFA：

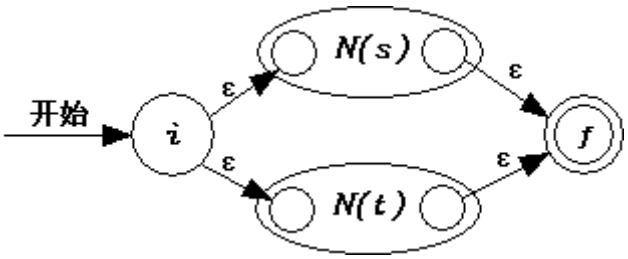


- **规则 2** 对于 Σ 的字母表中的元素 a，生成下面的 NFA：

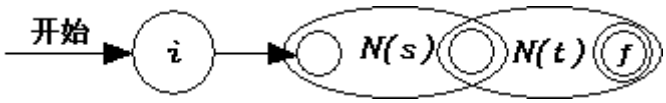


- **规则 3** 令正规表达式 s 和 t 的 NFA 分别为 N(s)和 N(t)：

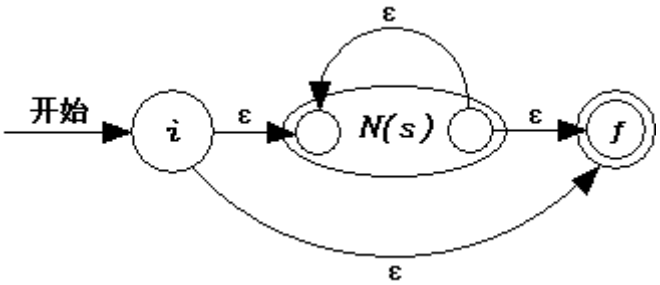
1. 对于 s|t，按照以下的方式生成 NFA N(s|t)：



2. 对于 st，按照以下的方式生成 NFA N(st)：



3. 对于 s^*，按照以下的方式生成 NFA N(s^*)：



4. 对于(s)，使用 s 本身的 NFA N(s)

3.2 NFA 到 DFA 的转化

输入： NFA N

输出： 能够接受与 N 相同语言的 DFA D

方法： 为 D 构造对应的状态迁移表 DTrans。DFA 的各个状态为 NFA 的状态集合，对于每一个输入符号，D 模拟 N 中可能的状态迁移。

定义转换操作如下表所示：

操作名称	描述
ε-closure(s)	从 NFA 的状态 s 出发，仅通过 ε 迁移能够到达的 NFA 的状态集合
ε-closure(T)	从 T 中包含的某个 NFA 的状态 s 出发，仅通过 ε 迁移能够到达的 NFA 的状态集合
move(T, a)	从 T 中包含的某个 NFA 的状态 s 出发，通过输入符号 a 迁移能够到达的 NFA 的状态集合

对于从 NFA 到 DFA 的转化，我们涉及到如下两个算法，即构造 NFA N 的状态子集是算法，以及计算 ε-closure(T)的算法，算法的伪代码可以表示如下：

1. 构造 NFA N 的状态 K 的子集算法：

```
令 DStates 中仅包含 ε-closure(s)，并设置状态为未标记;
while DStates 中包含未标记的状态 T
    do begin
        标记 T;
        for 各输入记号 a
    do begin
         U = ε-closure(move(T, a));
         if U 不在 DStates 中
          then 将 U 追加到 DStates 中，设置状态为未标记;
               DTrans [T, a] = U;
        end
 end
```


2. ε -closure(T)的计算方法如下：

```
将 T 中的所有状态入栈;
 设置 ε-closure(T)的初始值为 T;
 while 栈非空
  do begin
      从栈顶取出元素 t;
      for 从 t 出发以 ε 为边能够到达的各个状态 u do
       if u 不在 ε-closure(T)中
       then begin
            将 u 追加到 ε-closure(T)中;
             将 u 入栈;
             end
      end
```