1

语言之间的运算　给定两个语言 L, M, 语言是字符串的集合. 语言之间有乘法, 幂, 闭包等运算. 注意到乘法仅仅是一个称呼.

定义 1.1 (运算)**.** 下面列出四种运算.

$$L \cdot M \equiv \{w \mid w = xy, x \in L, y \in M\}$$
$$L + M \equiv L \cup M$$
$$L^2 = L \cdot L \text{ 特别地}, L^0 = \{\epsilon\}$$
$$L^* = \bigcup_{i=0}^{\infty} L^i$$

正则表达式　的递归定义.

定义 1.2. 分为基础部分, 归纳部分. 基础部分: 空集是一个正则表达式, 随后 ϵ 也是一个正则表达式. 对于任意一个符号 a , 其也是一个正则表达式. 归纳部分: E_1, E_2 都是正则表达式, 那么 $E_1 + E_2$ 也是正则表达式, $+$ 表示或; E_1E_2 也是正则表达式; E_1^*, E_2^* 也是正则表达式; 括号用来表示运算的顺序.

Example 1.3. *下面正则表达式的语言是什么?*

$$1 + 01^* \tag{1}$$

Example 1.4.

$$(a + b)^*(a + bb) \tag{2}$$

a, b 组成的, 以 a 或者 bb 结尾的字符串.

正则表达式和 DFA　正如我们前面已经了解到的那样, 正则表达式和 DFA 的描述能力实际上是一样的, 也就是说 $L(E) = L(D)$, 我们将证明这种等价性. 我们分为两个方向, 给定一个正则表达式, 构造一个 DFA , 和给定一个 DFA 构造一个正则表达式.

给定 DFA 构造正则表达式 有两种方法: 1. 递归法; 2. 状态消除法.

递归法 给 DFA 进行编号, 定义 $R_{i,j}^{k}$, 其表示的是, 由 i 到 j 的, 中间经过的状态的编号不超过 k 的, 一个字符串的集合.

随后, 我们要遍历 $k=0,1,2,\ldots$. 考虑其初始状态, $k=0$ 的时候, 分两种情况讨论 $i=j, i\neq j$ 讨论.

其后, 开始遍历, 对于 $k>1$, 我们有

$$R_{i,j}^{(k)} = R_{i,j}^{(k-1)} + R_{i,k}^{(k-1)}(R_{k,k}^{(k-1)})^{*}R_{k,j}^{(k-1)} \tag{3}$$

其中 $R_{i,j}^{(k-1)}$ 是指不经过 k 的路径[1]. 我们能够看出, 这个递归方法是眼熟的, 这不是 tm 的动态规划吗? 随后就好理解了.

故, 我们和 DFA 等价的表达式是

$$\sum_{j\in F} R_{1,j}^{(n)} \tag{4}$$

Example 1.5. 见 *ppt* , *流程还是比较麻烦的.*

状态消除法 我超, 这个有点难搞, 不会画图, 反正也不是很难, 要不就直接看 ppt 得了.

[1]路径实际上是一个字符串!