

# Chapter 3

## 正则表达式和正则语言

### 3.1 正则表达式

自动机通过识别来定义语言, 正则表达式通过规则产生语言 (或表示语言); 正则表达式所表示的语言与正则语言等价.

#### 3.1.1 语言的运算

如果  $L$  和  $M$  是两个语言:

- (1)  $L \cup M$  为两个语言的并
- (2)  $LM$  为两个语言的连接
- (3)  $L^*$  为语言的 (克林) 闭包

#### 示例

若  $L = \{0, 11\}$ ,  $M = \{\varepsilon, 001\}$ , 那么

- (1)  $L \cup M = \{0, 11, \varepsilon, 001\}$
- (2)  $LM = \{0, 0001, 11, 11001\}$ ,  $ML = \{0, 11, 0010, 00111\}$
- (3)  $L^0 = \{\varepsilon\}$ ,  $L^1 = L = \{0, 11\}$ ,  $L^2 = \{00, 011, 110, 1111\}$ ,  
 $L^3 = \{000, 0011, 0110, 01111, 1100, 11011, 11110, 111111\}$ ,  $\dots$   
 $L^* = \bigcup_{i=0}^{\infty} L^i$

#### 示例

四则运算表达式的定义

- (1) 任何的数都是四则运算表达式;
- (2) 如果  $a$  和  $b$  是四则运算表达式, 那么  $a + b$ ,  $a - b$ ,  $a \times b$ ,  $a \div b$  和  $(a)$  都是四则运算表达式.

### 3.1.2 正则表达式的递归定义

设  $\Sigma$  为字母表, 则  $\Sigma$  上的正则表达式 (Regular Expression), 递归定义为:

- (1)  $\emptyset$  是一个正则表达式, 表示空语言;
- (2)  $\varepsilon$  是一个正则表达式, 表示语言  $\{\varepsilon\}$ ;
- (3)  $\Sigma$  中的任意字符  $a$ , 都是一个正则表达式, 分别表示语言  $\{a\}$ ;
- (4) 如果正则表达式  $r$  和  $s$  分别表示语言  $R$  和  $S$ , 则  $r + s$ ,  $rs$ ,  $r^*$  和  $(r)$  也是正则表达式, 分别表示语言  $R \cup S$ ,  $RS$ ,  $R^*$  和  $R$ .

此外  $r^+ = rr^*$  也称为正闭包, 显然  $r^* = r^+ + \varepsilon$ . 而且  $r^* = r^+$  当且仅当  $\varepsilon \in \mathbf{L}(r)$ .

#### 示例

给出正则表达式  $(a + b)^*(a + bb)$  定义的语言.

因为

- (1)  $a \rightarrow \{a\}$ ,  $b \rightarrow \{b\}$ ;
- (2)  $a + b \rightarrow \{a\} \cup \{b\} = \{a, b\}$ ,  $bb \rightarrow \{b\}\{b\} = \{bb\}$ ;
- (3)  $a + bb \rightarrow \{a\} \cup \{bb\} = \{a, bb\}$
- (4)  $(a + b)^* = \{a, b\}^*$
- (5)  $(a + b)^*(a + bb) \rightarrow \{a, b\}^*\{a, bb\} = \{a, bb, aa, abb, ba, bbb, \dots\}$

所以  $\mathbf{L}((a + b)^*(a + bb)) = \{w \mid w \text{ 由 } a \text{ 和 } b \text{ 组成, 仅以 } a \text{ 或 } bb \text{ 结尾.}\}$

#### 示例

给出正则表达式  $(aa)^*(bb)^*b$  定义的语言.

$$\mathbf{L}((aa)^*(bb)^*b) = (\{a\}\{a\})^*(\{b\}\{b\})^*\{b\} = \{aa\}^*\{bb\}^*\{b\} = \{a^{2n}b^{2m+1} \mid n \geq 0, m \geq 0\}$$

#### 示例

Design regular expression for  $L = \{w \mid w \in \{0, 1\}^* \text{ and } w \text{ contains } 01.\}$

$$(0 + 1)^*01(0 + 1)^*$$

#### 示例

Design regular expression for

$L = \{w \mid w \text{ consists of } 0\text{'s and } 1\text{'s, and the third symbol from the right end is } 1.\}$

$$(0 + 1)^*1(0 + 1)(0 + 1)$$

#### 示例

Design regular expression for  $L = \{w \mid w \in 0, 1^* \text{ and } w \text{ has no pair of consecutive } 0\text{'s.}\}$

$$1^*(011^*)^*(0 + \varepsilon) \text{ 或 } (1 + 01)^*(0 + \varepsilon)$$

#### 示例

$00$  表示语言  $\{00\}$ .  $(0 + 1)^*$  表示任意 0 和 1 构成的串.  $(0 + 1)^*00(0 + 1)^*$  表示至少有两个连续的 0 的串.  $(0 + \varepsilon)(1 + 10)^*$  表示没有两个连续 0 的串.

### 3.1.3 运算符的优先级

正则表达式的三种运算: “加” (+), “连接” ( $\cdot$  一般省略) 和 “星” (\*).

正则表达式中运算符的优先级: 括号的优先级最高, 但括号本身并不是运算

- (1) 首先, “星” 优先级最高:  $r^*$
- (2) 其次, “连接”:  $rs, r \cdot s$
- (3) 最后, “加” 优先级最低:  $r + s$

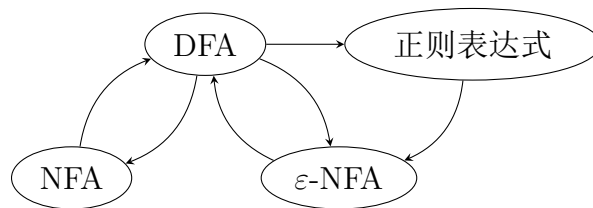
示例

$$01^* + 1 = (0(1^*)) + 1$$

## 3.2 有穷自动机和正则表达式

### 3.2.1 正则语言的表示

DFA, NFA,  $\varepsilon$ -NFA 和正则表达式在表示语言的能力上是等价的.

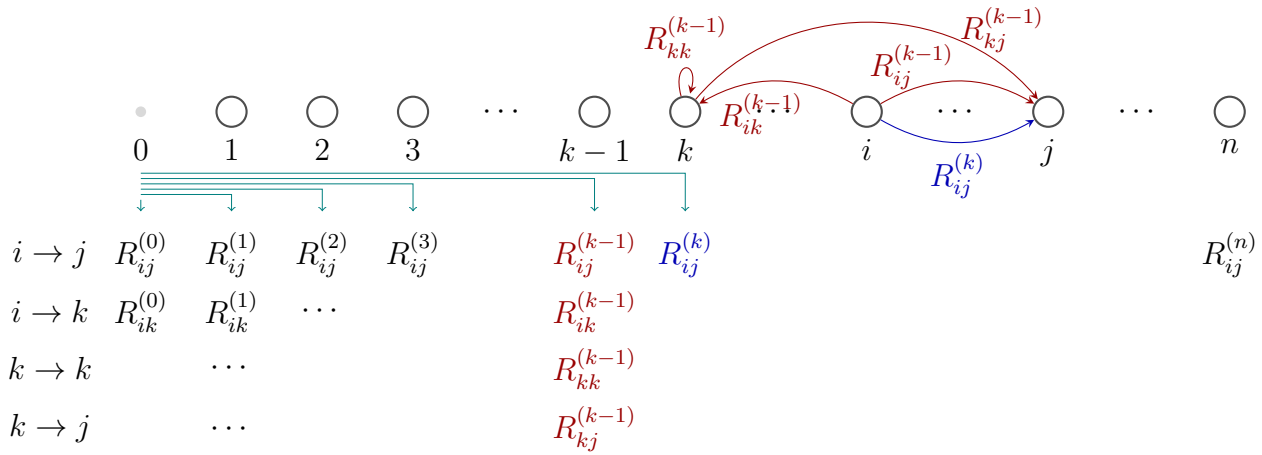


### 3.2.2 DFA $\Rightarrow$ RE, 递归构造 $R_{ij}^{(k)}$

**定理 1.** 如果  $L = \mathbf{L}(A)$  是某个 DFA  $A$  的语言, 则有一个正则表达式  $R$ , 且  $L = \mathbf{L}(R)$ .

设 DFA  $A$  的状态共有  $n$  个, 若为每个结点编号, DFA  $A$  的状态可表示为  $\{1, 2, \dots, n\}$ . 设  $R_{ij}^{(k)} = \{x \mid \hat{\delta}(q_i, x) = q_j\}$ . 也就是说,  $R_{ij}^{(k)}$  所有那样的字符串的集合 (也就是正则表达式), 即它能够使有穷自动机从状态  $q_i$  到达状态  $q_j$ , 而不通过编号高于  $k$  的任何状态. 正则表达式  $R_{ij}^{(k)}$  表示, 在结点  $i$  到  $j$  的全部路径中, 路径所经过的结点不超过  $k$  的全部路径. 但不包括起点  $i$  和终点  $j$ , 即起点和终点可以和  $k$  无关.

如果 1 是开始结点, 则该 DFA 的正则表达式就是  $\cup_{j \in F} R_{1j}^{(n)}$ .



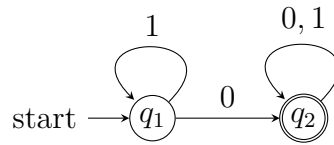
递推关系为:

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} \cup R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^*R_{kj}^{(k-1)}$$

$$R_{ij}^{(0)} = \begin{cases} \{a | \delta(q_i, a) = q_j\} & i \neq j \\ \{a | \delta(q_i, a) = q_j\} \cup \{\varepsilon\} & i = j \end{cases}$$

### 示例

例 1. 把下图所示 DFA 转换为正则表达式. 这个 DFA 接受至少含有一个 0 的串.



首先  $R_{ij}^{(0)}$ :

	应用公式 $R_{ij}^{(0)}$
$R_{11}^{(0)}$	$\varepsilon + 1$
$R_{12}^{(0)}$	$0$
$R_{21}^{(0)}$	$\emptyset$
$R_{22}^{(0)}$	$\varepsilon + 0 + 1$

简化规则:

$$(\varepsilon + R)^* = R^*$$

$$R + RS^* = RS^*$$

$$\emptyset R = R\emptyset = \emptyset \text{ (零元)}$$

$$\emptyset + R = R + \emptyset = R \text{ (单位元)}$$

计算  $R_{ij}^{(1)}$ :

	应用公式 $R_{ij}^{(k)}$	化简
$R_{11}^{(1)}$	$\varepsilon + 1 + (\varepsilon + 1)(\varepsilon + 1)^*(\varepsilon + 1)$	$1^*$
$R_{12}^{(1)}$	$0 + (\varepsilon + 1)(\varepsilon + 1)^*0$	$1^*0$
$R_{21}^{(1)}$	$\emptyset + \emptyset(\varepsilon + 1)^*(\varepsilon + 1)$	$\emptyset$
$R_{22}^{(1)}$	$\varepsilon + 0 + 1 + \emptyset(\varepsilon + 1)^*0$	$\varepsilon + 0 + 1$

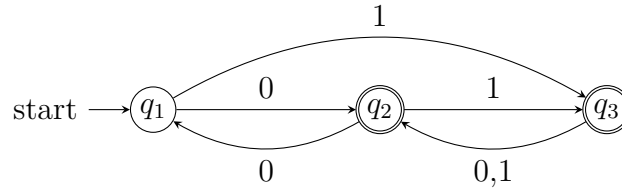
计算  $R_{ij}^{(2)}$ :

	应用公式 $R_{ij}^{(k)}$	化简
$R_{11}^{(2)}$	$1^* + 1^*0(\varepsilon + 0 + 1)^*\emptyset$	$1^*$
$R_{12}^{(2)}$	$1^*0 + 1^*0(\varepsilon + 0 + 1)^*(\varepsilon + 0 + 1)$	$1^*0(0 + 1)^*$
$R_{21}^{(2)}$	$\emptyset + (\varepsilon + 0 + 1)(\varepsilon + 0 + 1)^*\emptyset$	$\emptyset$
$R_{22}^{(2)}$	$\varepsilon + 0 + 1 + (\varepsilon + 0 + 1)(\varepsilon + 0 + 1)^*(\varepsilon + 0 + 1)$	$(0 + 1)^*$

由于 1 是开始状态, 2 是唯一的接受状态, 结点个数是 2, 所以 DFA 的正则表达式只需要  $R_{12}^{(2)} = 1^*0(0+1)^*$ .

### 示例

例 2. 把下图所示 DFA 转换为正则表达式.



得:

	$k = 0$	$k = 1$	$k = 2$
$R_{11}^{(k)}$	$\varepsilon$	$\varepsilon$	$(00)^*$
$R_{12}^{(k)}$	0	0	$0(00)^*$
$R_{13}^{(k)}$	1	1	$0^*1$
$R_{21}^{(k)}$	0	0	$0(00)^*$
$R_{22}^{(k)}$	$\varepsilon$	$\varepsilon + 00$	$(00)^*$
$R_{23}^{(k)}$	1	$1 + 01$	$0^*1$
$R_{31}^{(k)}$	$\emptyset$	$\emptyset$	$(0+1)(00)^*0$
$R_{32}^{(k)}$	$0+1$	$0+1$	$(0+1)(00)^*$
$R_{33}^{(k)}$	$\varepsilon$	$\varepsilon$	$\varepsilon + (0+1)0^*1$

状态 2 和 3 是接受状态, 所以仅计算:

$$\begin{aligned}
 R_{12}^{(3)} &= R_{12}^{(2)} + R_{13}^{(2)}(R_{33}^{(2)})^*R_{32}^{(2)} \\
 &= 0^*1(\varepsilon + (0+1)0^*1)^*(0+1)(00)^* + 0(00)^* \\
 &= 0^*1((0+1)0^*1)^*(0+1)(00)^* + 0(00)^*
 \end{aligned}$$

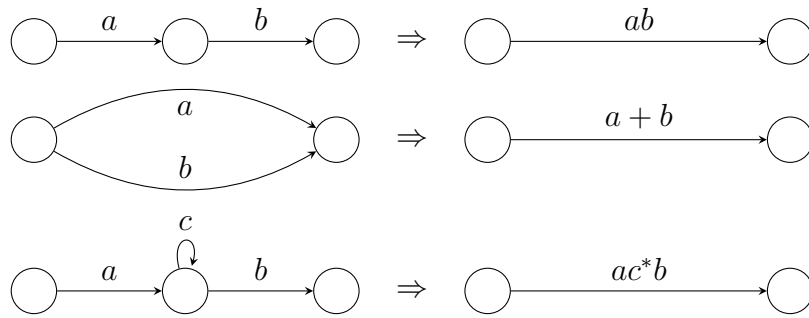
$$\begin{aligned}
 R_{13}^{(3)} &= R_{13}^{(2)} + R_{13}^{(2)}(R_{33}^{(2)})^*R_{33}^{(2)} \\
 &= 0^*1(\varepsilon + (0+1)0^*1)^*(\varepsilon + (0+1)0^*1) + 0^*1 \\
 &= 0^*1((0+1)0^*1)^*
 \end{aligned}$$

则 DFA 的正则表达式为:

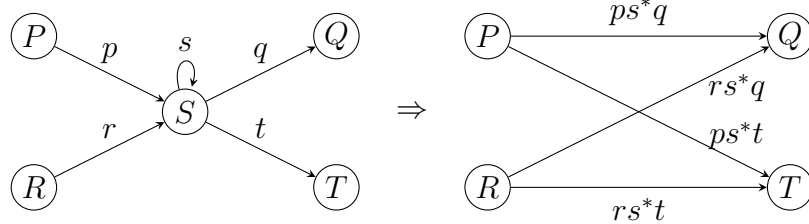
$$R_{12}^{(3)} + R_{13}^{(3)} = 0^*1((0+1)0^*1)^*(\varepsilon + (0+1)(00)^*) + 0(00)^*.$$

### 3.2.3 DFA $\Rightarrow$ RE, 状态消除

从有穷自动机中删除状态, 并使用新的路径替换被删除的路径, 在新路径上构造新的正则表达式, 产生一个等价的自动机. 那么, 删除一个状态, 可能有的最简单的情况如下三种:

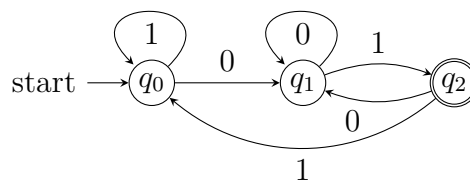


更一般的情况, 如下图, 要为被删除的状态  $S$  的每个“入”和“出”路径的组合, 补一条等价的新路径, 并用新的正则表达式表示:

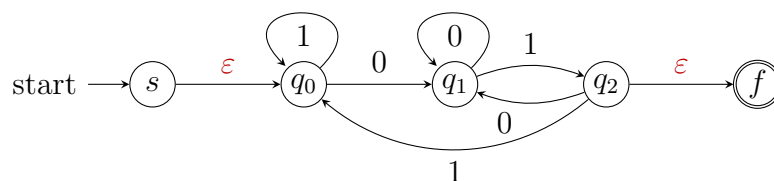


### 示例

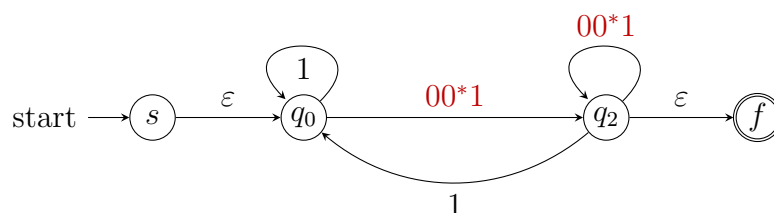
状态转移图如下, 通过状态消除构造正则表达式.



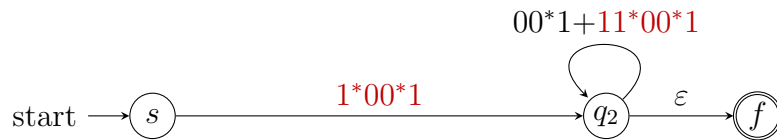
增加新的开始 ( $s$ ) 和结束状态 ( $f$ ) 及空转移:



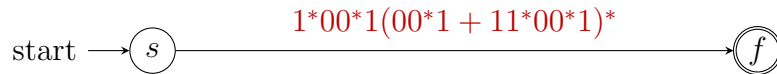
消除状态  $q_1$ , 需要增加  $q_0 \rightarrow q_2$  和  $q_2 \rightarrow q_2$  两条路径:



消除状态  $q_0$ , 需要增加  $s \rightarrow q_2$  和  $q_2 \rightarrow q_2$  两条路径:



消除状态  $q_2$ , 需要增加  $s \rightarrow f$  一条路径:

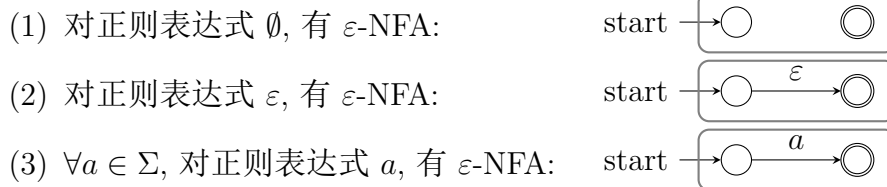


### 3.2.4 RE $\Rightarrow$ DFA, 构造 $\varepsilon$ -NFA

**定理 2.** 每个有正则表达式定义的语言都可以由有穷自动机识别.

命题: 如果  $R$  是正则表达式, 则存在一个  $\varepsilon$ -NFA  $E$ , 使  $L(E) = L(R)$ , 且  $E$  满足 (1) 仅有一个接收状态; (2) 没有进入开始状态的边; (3) 没有离开接受状态的边.

则通过归纳法, 可以证明. 其中归纳基础为

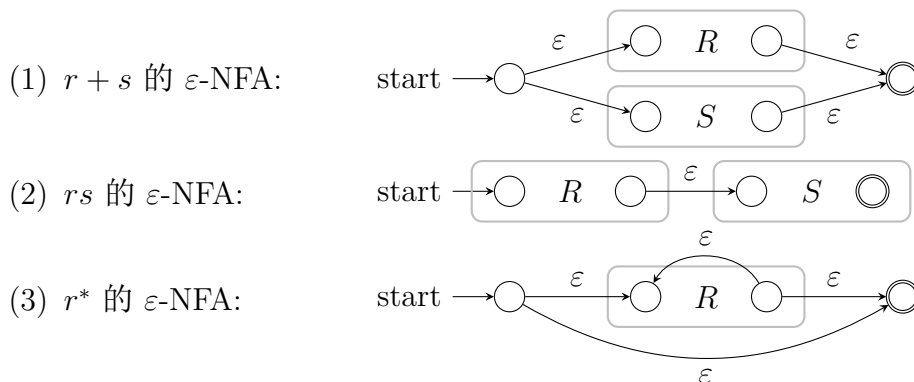


归纳递推为:

若  $r$  和  $s$  为正则表达式, 则它们对应的  $\varepsilon$ -NFA 分别为  $R$  和  $S$

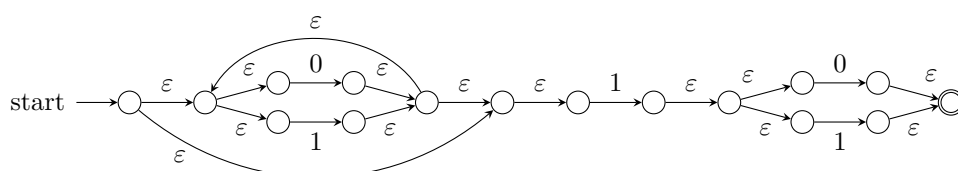


则正则表达式  $r + s$ ,  $rs$  和  $r^*$ , 则可以分别由  $R$  和  $S$  构造如下:



**示例**

正则表达式  $(0 + 1)^*1(0 + 1)$  构造为  $\varepsilon$ -NFA.



## 3.3 正则表达式的代数定律

在保持表达的语言不变的前提下, 对表达式进行化简的代数规则.

### 3.3.1 结合律 (Associativity) 和交换律 (Commutativity)

- $L + M = M + L$  并的交换律
- $(L + M) + N = L + (M + N)$  并的结合律
- $(LM)N = L(MN)$  连接的结合律
- 连接不满足交换律  $LM \neq ML$

### 3.3.2 单位元 (Identities) 与零元 (Annihilators)

- $\emptyset + L = L + \emptyset = L$  并运算的单位元  $\emptyset$
- $\varepsilon L = L\varepsilon = L$  连接运算的单位元  $\varepsilon$
- $\emptyset L = L\emptyset = \emptyset$  连接运算的零元

### 3.3.3 分配率 (Distributive Laws)

- $L(M + N) = LM + LN$  连接对并满足左分配律
- $(M + N)L = ML + NL$  连接对并满足右分配律

示例

$$0 + 01^* = 0\varepsilon + 01^* = 0(\varepsilon + 1^*) = 01^*$$

### 3.3.4 幂等律 (The Idempotent Law)

- $L + L = L$  并的幂等律

### 3.3.5 有关闭包的定律

- $(L^*)^* = L^*$  对某语言的闭包再取闭包, 并不改变语言
- $\emptyset^* = \varepsilon$
- $\varepsilon^* = \varepsilon$  空串的连接仍然是空串
- $L^* = L^+ + \varepsilon$
- $L? = \varepsilon + L$



### 3.3.6 发现正则表达式的定律

- $(L + M)^* = (L^* M^*)^*$
- $L^* L^* = L^*$
- $(\varepsilon + L)^* = L^*$