

# Chapter 3

## 正则表达式

### 3.1 正则表达式

- 有穷自动机
  - 通过机器装置描述正则语言
  - 用计算机编写相应算法, 易于实现
- 正则表达式
  - 通过表达式描述正则语言, 代数表示方法, 使用方便
  - 应用广泛
    - \* `grep` 工具 (Global Regular Expression and Print)
    - \* Emacs / Vim 文本编辑器
    - \* `lex` / `flex` 词法分析器
    - \* 各种程序设计语言 Python / Perl / Haskell / ...

#### 3.1.1 语言的运算

设  $L$  和  $M$  是两个语言, 那么

并  $L \cup M = \{w \mid w \in L \text{ 或 } w \in M\}$

连接  $L \cdot M = \{w \mid w = xy, x \in L \text{ 且 } y \in M\}$

幂 
$$\begin{aligned} L^0 &= \{\varepsilon\} \\ L^1 &= L \\ L^n &= L^{n-1} \cdot L \end{aligned}$$

克林闭包

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

例 1. 若有语言  $L = \{0, 11\}$  和  $M = \{\varepsilon, 001\}$ , 那么

$$\begin{array}{ll} L \cup M = & L^0 = \\ LM = & L^1 = \\ ML = & L^2 = \end{array}$$

例 2. 对于空语言  $\emptyset$

$$\begin{array}{l} \emptyset^0 = \{\varepsilon\} \\ \forall n \geq 1, \emptyset^n = \emptyset \\ \emptyset^* = \{\varepsilon\} \end{array}$$

四则运算表达式的递归定义:

1. 任何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

$$a + b, a - b, a \times b, a \div b \text{ 和 } (a)$$

都是四则运算表达式.

### 3.1.2 正则表达式的递归定义

定义. 如果  $\Sigma$  为字母表, 则  $\Sigma$  上的正则表达式递归定义为:

1.  $\emptyset$  是一个正则表达式, 表示空语言;  
 $\varepsilon$  是一个正则表达式, 表示语言  $\{\varepsilon\}$ ;  
 $\forall a \in \Sigma, a$  是一个正则表达式, 表示语言  $\{a\}$ ;
2. 如果正则表达式  $r$  和  $s$  分别表示语言  $R$  和  $S$ , 那么

$$r + s, rs, r^* \text{ 和 } (r)$$

都是正则表达式, 分别表示语言

$$R \cup S, R \cdot S, R^* \text{ 和 } R.$$

此外正闭包定义为  $r^+ = rr^*$ , 显然  $r^* = r^+ + \varepsilon$ . 而且  $r^* = r^+$  当且仅当  $\varepsilon \in L(r)$ .

### 3.1.3 运算符的优先级

正则表达式中三种运算以及括号的优先级:

1. 首先, “括号” 优先级最高;
2. 其次, “星” 运算:  $\mathbf{r}^*$ ;
3. 然后, “连接” 运算:  $\mathbf{rs}$ ,  $\mathbf{r} \cdot \mathbf{s}$ ;
4. 最后, “加” 最低:  $\mathbf{r} + \mathbf{s}$ ,  $\mathbf{r} \cup \mathbf{s}$ ;

例 3.

$$\begin{aligned}
 \mathbf{1} + \mathbf{01}^* &= \mathbf{1} + (\mathbf{0(1^*)}) \\
 &\neq \mathbf{1} + (\mathbf{01})^* \\
 &\neq (\mathbf{1} + \mathbf{01})^* \\
 &\neq (\mathbf{1} + \mathbf{0})\mathbf{1}^*
 \end{aligned}$$

### 3.1.4 正则表达式示例

例 4.

$E$	$L(E)$
$\mathbf{a} + \mathbf{b}$	$L(\mathbf{a}) \cup L(\mathbf{b}) = \{a\} \cup \{b\} = \{a, b\}$
$\mathbf{bb}$	$L(\mathbf{b}) \cdot L(\mathbf{b}) = \{b\} \cdot \{b\} = \{bb\}$
$(\mathbf{a} + \mathbf{b})(\mathbf{a} + \mathbf{b})$	$\{a, b\}\{a, b\} = \{aa, ab, ba, bb\}$
$(\mathbf{a} + \mathbf{b})^*(\mathbf{a} + \mathbf{bb})$	$\{a, b\}^*\{a, bb\} = \{a, b\}^*\{a\} \cup \{a, b\}^*\{bb\} = \{w \in \{a, b\}^* \mid w \text{ 仅以 } a \text{ 或 } bb \text{ 结尾.}\}$
$\mathbf{1} + (\mathbf{01})^*$	$\{1, \varepsilon, 01, 0101, 010101, \dots\}$
$(\mathbf{0} + \mathbf{1})^*\mathbf{01}(\mathbf{0} + \mathbf{1})^*$	$\{x01y \mid x, y \in \{0, 1\}^*\}$

例 5. 给出正则表达式  $(\mathbf{aa})^*(\mathbf{bb})^*\mathbf{b}$  定义的语言.

$$\begin{aligned}
 L((\mathbf{aa})^*(\mathbf{bb})^*\mathbf{b}) &= L((\mathbf{aa})^*) \cdot L((\mathbf{bb})^*) \cdot L(\mathbf{b}) \\
 &= (\{a\}\{a\})^*(\{b\}\{b\})^*\{b\}
 \end{aligned}$$

$$\begin{aligned}
 &= \{a^2\}^* \{b^2\}^* \{b\} \\
 &= \{a^{2n} b^{2m+1} \mid n \geq 0, m \geq 0\}
 \end{aligned}$$

例 6. Design regular expression for  $L = \{w \mid w \text{ consists of 0's and 1's, and the third symbol from the right end is 1.}\}$

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

例 7. Design regular expression for  $L = \{w \mid w \in \{0, 1\}^* \text{ and } w \text{ has no pair of consecutive 0's.}\}$

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

例. Write a regular expression for the set of strings that consist of alternating 0's and 1's.

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

例. Write a regular expression for  $L = \{w \in \{0, 1\}^* \mid 0 \text{ and } 1 \text{ alternate in } w\}$ .

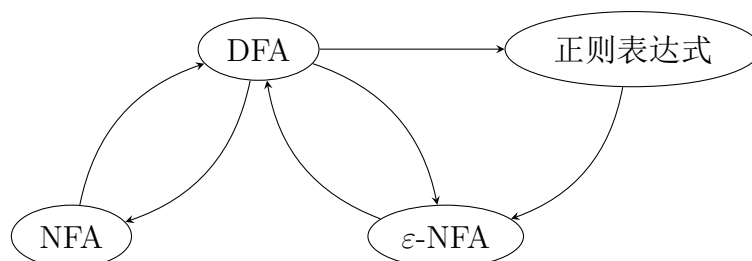
例. Find a regular expression for the set  $\{a^n b^m \mid (n + m) \text{ is odd}\}$ .

例. Give regular expression for the complement of  $L = \{a^n b^m \mid n \geq 3, m \leq 4\}$ .

例. Write a regular expression for the set of all C real numbers.

## 3.2 有穷自动机和正则表达式

DFA, NFA,  $\varepsilon$ -NFA 和正则表达式的等价性



### 3.2.1 由 DFA 到正则表达式, 递归表达式法

定理 3. 若  $L = L(A)$  是某 DFA  $A$  的语言, 那么存在正则表达式  $R$  满足  $L = L(R)$ .

证明: 对 DFA  $A$  的状态编号, 令 1 为开始状态, 即

$$A = (\{1, 2, \dots, n\}, \Sigma, \delta, 1, F),$$

设正则表达式  $R_{ij}^{(k)}$  表示从  $i$  到  $j$  但中间节点不超过  $k$  全部路径的字符串集:

$$R_{ij}^{(k)} = \{x \mid \hat{\delta}(i, x) = j, x \text{ 经过的状态除两端外都不超过 } k\}.$$



那么与  $A = (\{1, 2, \dots, n\}, \Sigma, \delta, 1, F)$  等价的正则表达式为

$$\bigcup_{j \in F} R_{1j}^{(n)}$$

且递归式为

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

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

下面对  $k$  归纳, 证明可用以上递归式求得  $R_{ij}^{(k)}$ .

归纳基础: 当  $i \neq j, k = 0$  时, 即  $i$  到  $j$  没经过任何中间节点

- 没有  $i$  到  $j$  的状态转移

$$\begin{array}{ccc} \textcircled{i} & & \textcircled{j} \end{array} \quad R_{ij}^{(0)} = \emptyset$$

- 有一个  $i$  到  $j$  的状态转移

$$\textcircled{i} \xrightarrow{a} \textcircled{j} \quad R_{ij}^{(0)} = a$$

- 有多个  $i$  到  $j$  的状态转移

$$\begin{array}{ccc} \textcircled{i} & \begin{array}{c} \xrightarrow{a_1} \\ \cdots \\ \xrightarrow{a_t} \end{array} & \textcircled{j} \end{array} \quad R_{ij}^{(0)} = a_1 + a_2 + \cdots + a_t$$

归纳基础 (续): 当  $i = j, k = 0$  时, 即从  $i$  到自身没经过任何中间节点

- 状态  $i$  没有到自己的转移



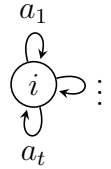
$$R_{ii}^{(0)} = \epsilon$$

- 状态  $i$  有一个到自身的转移



$$R_{ii}^{(0)} = a + \epsilon$$

- 状态  $i$  有多个到自身的转移



$$R_{ii}^{(0)} = a_1 + a_2 + \dots + a_t + \epsilon$$

归纳假设: 已知  $R_{ij}^{(k-1)}$  是从  $i$  到  $j$  但中间节点不超过  $k-1$  的全部路径, 同理已知  $R_{ik}^{(k-1)}$ ,  $R_{kk}^{(k-1)}$  和  $R_{kj}^{(k-1)}$ .

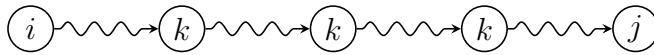
归纳递推: 那么  $R_{ij}^{(k)}$  中全部路径, 可用节点  $k$  分为两部分

- 从  $i$  到  $j$  不经过  $k$  的



$$R_{ij}^{(k)} = R_{ij}^{(k-1)}$$

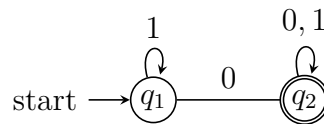
- 从  $i$  到  $j$  经过  $k$  的



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

因此  $R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$ .

例 8. 将如图 DFA 转换为正则表达式.



- 计算  $R_{ij}^{(0)}$

$R_{ij}^{(k)}$	$k = 0$
$R_{11}^{(0)}$	$\epsilon + 1$
$R_{12}^{(0)}$	$0$
$R_{21}^{(0)}$	$\emptyset$
$R_{22}^{(0)}$	$\epsilon + 0 + 1$

- 计算  $R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)}(R_{11}^{(0)})^*R_{1j}^{(0)}$

$R_{ij}^{(k)}$	$k = 0$	$R_{ij}^{(k)}$	$k = 1$
$R_{11}^{(0)}$	$\varepsilon + \mathbf{1}$	$R_{11}^{(1)}$	$(\varepsilon + \mathbf{1}) + (\varepsilon + \mathbf{1})(\varepsilon + \mathbf{1})^*(\varepsilon + \mathbf{1})$
$R_{12}^{(0)}$	$\mathbf{0}$	$R_{12}^{(1)}$	$\mathbf{0} + (\varepsilon + \mathbf{1})(\varepsilon + \mathbf{1})^*\mathbf{0}$
$R_{21}^{(0)}$	$\emptyset$	$R_{21}^{(1)}$	$\emptyset + \emptyset(\varepsilon + \mathbf{1})^*(\varepsilon + \mathbf{1})$
$R_{22}^{(0)}$	$\varepsilon + \mathbf{0} + \mathbf{1}$	$R_{22}^{(1)}$	$\varepsilon + \mathbf{0} + \mathbf{1} + \emptyset(\varepsilon + \mathbf{1})^*\mathbf{0}$

- 几个基本的化简规则

如果  $\mathbf{r}$  和  $\mathbf{s}$  是两个正则表达式

$$(\varepsilon + \mathbf{r})^* = \mathbf{r}^*$$

$$(\varepsilon + \mathbf{r})\mathbf{r}^* = \mathbf{r}^*$$

$$\mathbf{r} + \mathbf{r}\mathbf{s}^* = \mathbf{r}\mathbf{s}^*$$

$$\emptyset\mathbf{r} = \mathbf{r}\emptyset = \emptyset \quad (\text{零元})$$

$$\emptyset + \mathbf{r} = \mathbf{r} + \emptyset = \mathbf{r} \quad (\text{单位元})$$

- 化简  $R_{ij}^{(1)}$

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

- 计算  $R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)}(R_{22}^{(1)})^*R_{2j}^{(1)}$

$R_{ij}^{(k)}$	$k = 1$	$R_{ij}^{(k)}$	$k = 2$
$R_{11}^{(1)}$	$\mathbf{1}^*$	$R_{11}^{(2)}$	$\mathbf{1}^* + \mathbf{1}^*\mathbf{0}(\varepsilon + \mathbf{0} + \mathbf{1})^*\emptyset$
$R_{12}^{(1)}$	$\mathbf{1}^*\mathbf{0}$	$R_{12}^{(2)}$	$\mathbf{1}^*\mathbf{0} + \mathbf{1}^*\mathbf{0}(\varepsilon + \mathbf{0} + \mathbf{1})^*(\varepsilon + \mathbf{0} + \mathbf{1})$
$R_{21}^{(1)}$	$\emptyset$	$R_{21}^{(2)}$	$\emptyset + (\varepsilon + \mathbf{0} + \mathbf{1})(\varepsilon + \mathbf{0} + \mathbf{1})^*\emptyset$
$R_{22}^{(1)}$	$\varepsilon + \mathbf{0} + \mathbf{1}$	$R_{22}^{(2)}$	$\varepsilon + \mathbf{0} + \mathbf{1} + (\varepsilon + \mathbf{0} + \mathbf{1})(\varepsilon + \mathbf{0} + \mathbf{1})^*(\varepsilon + \mathbf{0} + \mathbf{1})$

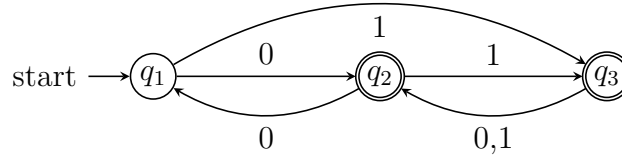
- 化简  $R_{ij}^{(2)}$

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

- 因只有  $q_2$  是接受状态, 所以该 DFA 正则表达式为

$$R_{12}^{(2)} = 1^*0(0 + 1)^*.$$

例 9. 将如图 DFA 转换为正则表达式.



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

仅状态 2 和 3 是接受状态:

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

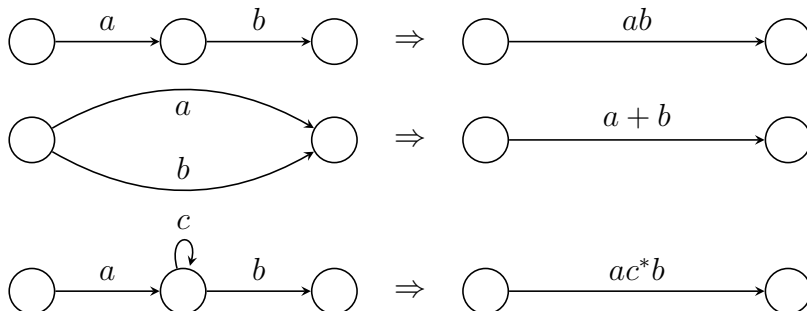
$$\begin{aligned}
 R_{13}^{(3)} &= R_{13}^{(2)} + R_{13}^{(2)}(R_{33}^{(2)})^*R_{33}^{(2)} \\
 &= 0^*1 + 0^*1(\varepsilon + (0 + 1)0^*1)^*(\varepsilon + (0 + 1)0^*1) \\
 &= 0^*1((0 + 1)0^*1)^*
 \end{aligned}$$

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

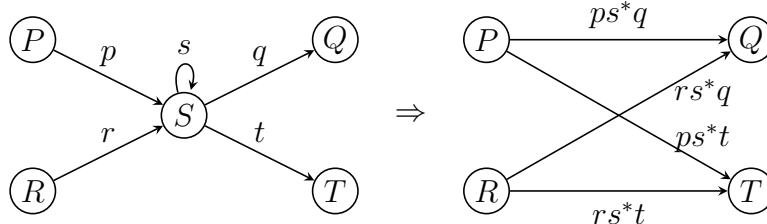


## 3.2.2 由 DFA 到正则表达式, 状态消除法

- 从 DFA 中逐个删除状态
- 用标记了正则表达式的新路径替换被删掉的路径
- 保持“自动机”等价.

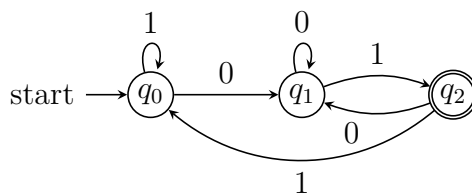


- 更一般的情况如图
- 若要删除状态  $S$ , 需添加相应路径

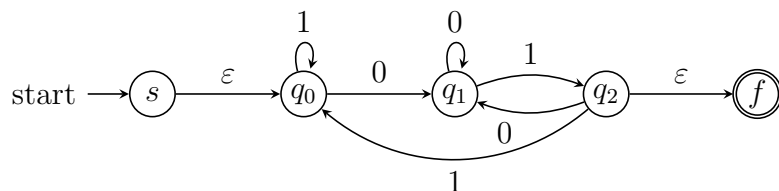


要为被删除的状态  $S$  的每个“入”和“出”路径的组合, 补一条等价的新路径, 结点上的循环用闭包表示. 保持新路径与被删掉的路径集合等价.

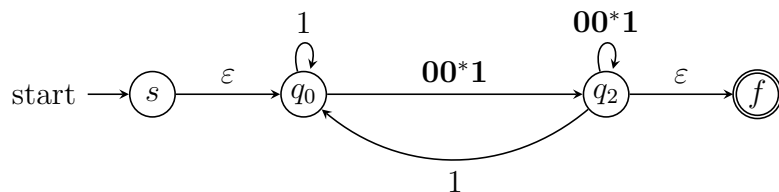
例 10. 利用状态消除法, 构造下图自动机的正则表达式.



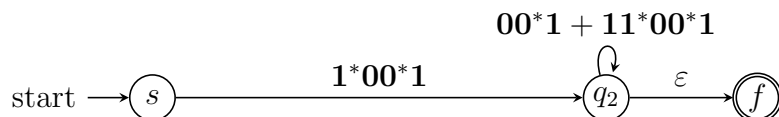
1. 利用空转移, 添加新的开始  $s$  和结束状态  $f$ :



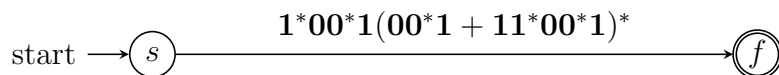
2. 消除状态  $q_1$ , 添加路径  $q_0 \rightarrow q_2$  和  $q_2 \rightarrow q_2$ :



3. 消除状态  $q_0$ , 添加路径  $s \rightarrow q_2$  和  $q_2 \rightarrow q_2$ :



4. 消除状态  $q_2$ , 添加路径  $s \rightarrow f$ :



5. 因此该自动机的正则表达式为

$$1^*00^*1(00^*1 + 11^*00^*1)^*.$$

### 3.2.3 由正则表达式到 $\varepsilon$ -NFA

**定理 4.** 正则表达式定义的语言, 都可被有穷自动机识别.

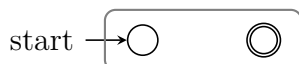
#### 由正则表达式构造 $\varepsilon$ -NFA

任何正则表达式  $e$ , 都存在与其等价的  $\varepsilon$ -NFA  $A$ , 即  $L(A) = L(e)$ , 并且  $A$  满足:

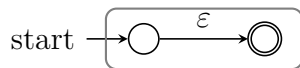
1. 仅有一个接收状态;
2. 没有进入开始状态的边;
3. 没有离开接受状态的边.

归纳基础:

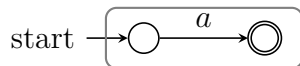
1. 对于  $\emptyset$ , 有  $\varepsilon$ -NFA:



2. 对于  $\epsilon$ , 有  $\epsilon$ -NFA:



3.  $\forall a \in \Sigma$ , 对于  $a$ , 有  $\epsilon$ -NFA:

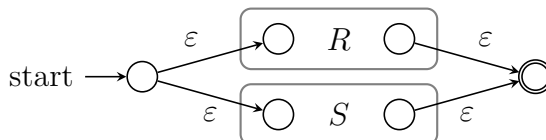


归纳递推: 若  $r$  和  $s$  为正则表达式, 则它们对应的  $\epsilon$ -NFA 分别为  $R$  和  $S$



则正则表达式  $r + s$ ,  $rs$  和  $r^*$ , 可由  $R$  和  $S$  分别构造如下:

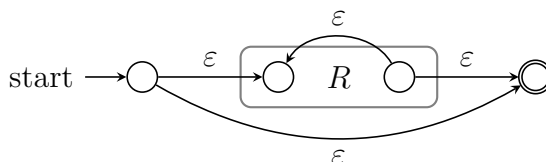
1. 对于  $r + s$ , 有  $\epsilon$ -NFA:



2. 对于  $rs$ , 有  $\epsilon$ -NFA:

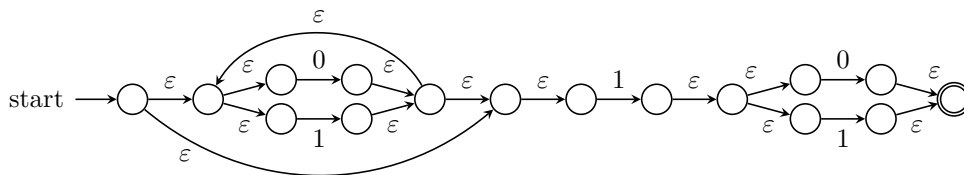


3. 对于  $r^*$ , 有  $\epsilon$ -NFA:



因此任何结构的正则表达式, 都可递归构造出等价的  $\epsilon$ -NFA.

例 11. 正则表达式  $(0 + 1)^*1(0 + 1)$  构造为  $\epsilon$ -NFA.



### 思考题

正则表达式到  $\epsilon$ -NFA 构造方法中的 3 个限制条件, 都有必要吗?

## 3.3 正则表达式的代数定律

### 3.3.1 基本的代数定律

定义. 含有变量的两个正则表达式, 如果以任意语言替换其变量, 二者所表示的语言仍然相同, 则称这两个正则表达式等价. 在这样的意义下, 正则表达式满足一些代数定律.

- 并运算

$$\begin{aligned}(L + M) + N &= L + (M + N) && \text{(结合律)} \\ L + M &= M + L && \text{(交换律)} \\ L + L &= L && \text{(幂等律)} \\ \emptyset + L &= L + \emptyset = L && \text{(单位元 } \emptyset \text{)}\end{aligned}$$

- 连接运算

$$\begin{aligned}(LM)N &= L(MN) && \text{(结合律)} \\ \epsilon L &= L\epsilon = L && \text{(单位元 } \epsilon \text{)} \\ \emptyset L &= L\emptyset = \emptyset && \text{(零元 } \emptyset \text{)} \\ LM &\neq ML\end{aligned}$$

- 分配率

$$\begin{aligned}L(M + N) &= LM + LN && \text{(左分配律)} \\ (M + N)L &= ML + NL && \text{(右分配律)}\end{aligned}$$

- 闭包运算

$$\begin{aligned}(L^*)^* &= L^* \\ \emptyset^* &= \epsilon \\ \epsilon^* &= \epsilon \\ L^* &= L^+ + \epsilon \\ (\epsilon + L)^* &= L^*\end{aligned}$$

### 3.3.2 发现与验证代数定律

#### 检验方法

要判断表达式  $E$  和  $F$  是否等价, 其中变量为  $L_1, \dots, L_n$ :

1. 将变量替换为具体表达式, 得正则表达式  $\mathbf{r}$  和  $\mathbf{s}$ , 例如, 替换  $L_i$  为  $\mathbf{a}_i$ ;
2. 判断  $\mathbf{L}(\mathbf{r}) \stackrel{?}{=} \mathbf{L}(\mathbf{s})$ , 如果相等则  $E = F$ , 否则  $E \neq F$ .

例 12. 判断  $(L + M)^* = (L^*M^*)^*$ .

1. 将  $L$  和  $M$  替换为  $\mathbf{a}$  和  $\mathbf{b}$ ;
2.  $(\mathbf{a} + \mathbf{b})^* \stackrel{?}{=} (\mathbf{a}^*\mathbf{b}^*)^*$ ;
3. 因为  $\mathbf{L}((\mathbf{a} + \mathbf{b})^*) = \mathbf{L}((\mathbf{a}^*\mathbf{b}^*)^*)$ ;
4. 所以  $(L + M)^* = (L^*M^*)^*$ .

例 13. 判断  $L + ML = (L + M)L$ .

1. 将  $L$  和  $M$  替换为  $\mathbf{a}$  和  $\mathbf{b}$ ;
2. 判断  $\mathbf{a} + \mathbf{ba} \stackrel{?}{=} (\mathbf{a} + \mathbf{b})\mathbf{a}$ ;
3. 因为  $aa \notin \mathbf{a} + \mathbf{ba}$  而  $aa \in (\mathbf{a} + \mathbf{b})\mathbf{a}$ ;
4. 所以  $\mathbf{a} + \mathbf{ba} \neq (\mathbf{a} + \mathbf{b})\mathbf{a}$ ;
5. 即  $L + ML \neq (L + M)L$ .

### 注意

这种方法仅限于判断正则表达式, 否则可能会发生错误.

例 14. 若用此方法判断  $L \cap M \cap N \stackrel{?}{=} L \cap M$ , 以  $\mathbf{a}, \mathbf{b}, \mathbf{c}$  替换  $L, M, N$ , 有

$$\{a\} \cap \{b\} \cap \{c\} = \emptyset = \{a\} \cap \{b\},$$

而显然

$$L \cap M \cap N \neq L \cap M.$$

例.

- $(L + M)^* = (L^*M^*)^*$
- $(\varepsilon + L)^* = L^*$
- $L^* \stackrel{?}{=} L^*L^*$

由  $\mathbf{a}^* = \mathbf{a}^*\mathbf{a}^*$  得  $L^* = L^*L^*$  成立.

- $(L + M)^* M \stackrel{?}{=} (L^* M)^*$

替换得  $(\mathbf{a} + \mathbf{b})^* \mathbf{b} \stackrel{?}{=} (\mathbf{a}^* \mathbf{b})^*$ , 因为  $\varepsilon \notin (\mathbf{a} + \mathbf{b})^* \mathbf{b}$  且  $\varepsilon \in (\mathbf{a}^* \mathbf{b})^*$ , 所以不相等.

- $(R + S)^* \stackrel{?}{=} R^* + S^*$

- $(RS + R)^* R \stackrel{?}{=} R(SR + R)^*$

- $(RS + R)^* RS \stackrel{?}{=} (RR^* S)^*$

- $(R + S)^* S \stackrel{?}{=} (R^* S)^*$