

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 (Union) $L \cup M = \{w \mid w \in L \text{ 或 } w \in M\}$

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

幂 (Power)

$$L^0 = \{\varepsilon\}$$
$$L^1 = L$$
$$L^n = L^{n-1} \cdot L$$

克林闭包 (Kleene Closure)

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

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

$$L \cup M = \quad L^0 =$$

$$LM = \quad L^1 =$$

$$ML = \quad L^2 =$$

例 2. 对于空语言 \emptyset

$$\emptyset^0 = \{\epsilon\}$$

$$\forall n \geq 1, \emptyset^n = \emptyset$$

$$\emptyset^* = \{\epsilon\}$$

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 Σ 为字母表, 则 Σ 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;
 ϵ 是一个正则表达式, 表示语言 $\{\epsilon\}$;
 $\forall a \in \Sigma, a$ 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 $r^* = r^+ + \epsilon$.

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.

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

3.1.4 正则表达式示例

例 4.

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

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

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

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

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

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

课堂练习.

Give regular expressions for each of the following languages over $\Sigma = \{0,1\}$.

- (1) All strings containing the substring 000.
- (2) All strings *not* containing the substring 000.

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

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

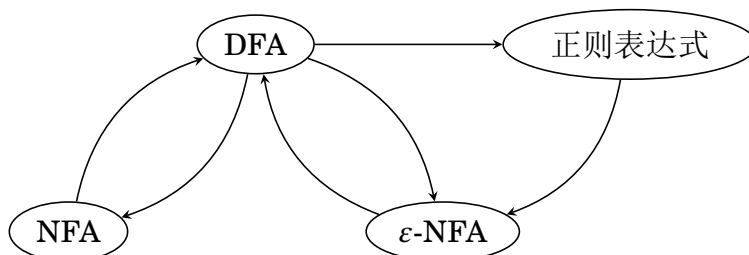
例. Find a regular expression for the set $\{a^n b^m \mid (n+m) \text{ is odd}\}$.

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, ε -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: 设 DFA A 的状态共有 n 个, 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:

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



也就是说, 正则表达式 $R_{ij}^{(k)}$ 是所有那样的字符串的集合, 它能够使有穷自动机从状态 i 到达状态 j , 但中间的路径上, 不经过编号高于 k 的任何状态.

如果 1 是开始结点, 对每个属于终态 F 的结点 j , 正则表达式 $R_{1j}^{(n)}$ 都是这个自动机所识别语言的一部分. 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

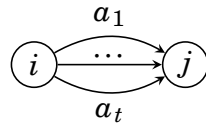
- 没有 i 到 j 的状态转移

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

- 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



$$R_{ij}^{(0)} = \mathbf{a}_1 + \mathbf{a}_2 + \cdots + \mathbf{a}_t$$

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



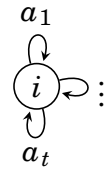
$$R_{ii}^{(0)} = \epsilon$$

- 状态 i 有一个到自身的转移



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

- 状态 i 有多个到自身的转移



$$R_{ii}^{(0)} = \mathbf{a}_1 + \mathbf{a}_2 + \cdots + \mathbf{a}_t + \epsilon$$

归纳假设: 假设已知 $R_{ij}^{(k-1)}, R_{ik}^{(k-1)}, R_{kk}^{(k-1)}$ 和 $R_{kj}^{(k-1)}$.

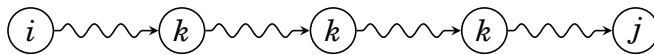
归纳递推: 那么 $R_{ij}^{(k)}$ 中全部路径, 可用节点 k 分为两部分

- 从 i 到 j 不经过 k 的



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

- 从 i 到 j 经过 k 的

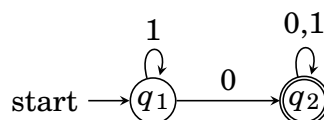


$$R_{ij}^{(k)} = R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$$

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 $R_{ij}^{(0)}$

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

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

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

- 几个基本的化简规则

如果 \mathbf{r} 和 \mathbf{s} 是两个正则表达式

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

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

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

$$\emptyset\mathbf{r} = \mathbf{r}\emptyset = \emptyset$$

零元

$$\emptyset + \mathbf{r} = \mathbf{r} + \emptyset = \mathbf{r}$$

单位元

- 化简 $R_{ij}^{(1)}$

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

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

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

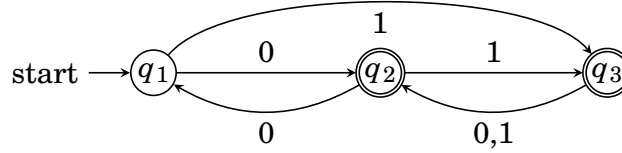
- 化简 $R_{ij}^{(2)}$

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

- 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.



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

仅状态 2 和 3 是接受状态:

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

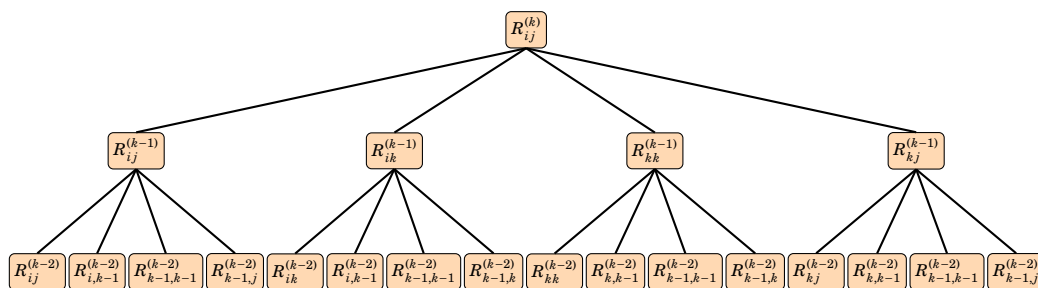
$$\begin{aligned}
&= 0(00)^* + 0^*1((0+1)0^*1)^*(0+1)(00)^* \\
R_{13}^{(3)} &= R_{13}^{(2)} + R_{13}^{(2)}(R_{33}^{(2)})^*R_{33}^{(2)} \\
&= 0^*1 + 0^*1(\epsilon + (0+1)0^*1)^*(\epsilon + (0+1)0^*1) \\
&= 0^*1((0+1)0^*1)^*
\end{aligned}$$

则 DFA 的正则表达式为:

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

分治 (Divide and Conquer) – 普遍且实用的递归求解方式

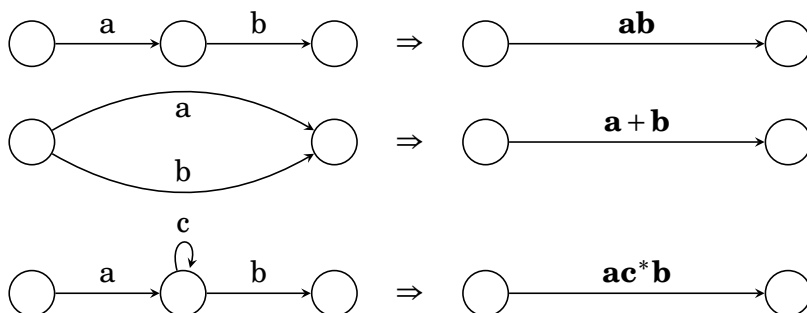
1. 将问题实例分解为子问题实例 – divide step
2. 子问题实例可递归解决
3. 将子问题实例合并可得到原问题实例 – conquer step



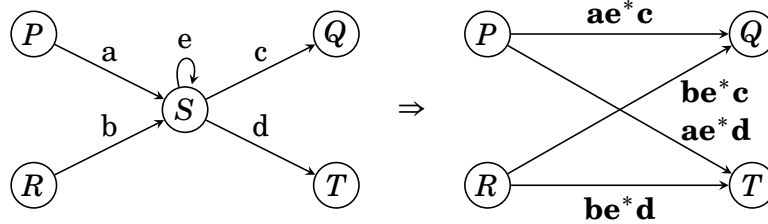
3.2.2 由 DFA 到正则表达式, 状态消除法

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, 在新路径上设计新的正则表达式, 产生一个等价的“自动机”。

- 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.

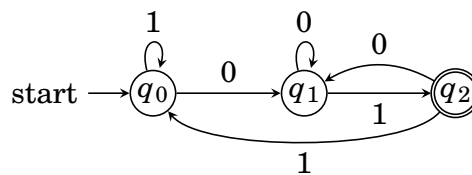


- 更一般的情况如图, 若要删除状态 S , 需添加相应路径

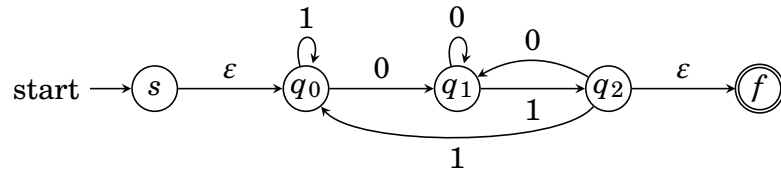


要为被删除的状态 S 的每个“入”和“出”路径的组合, 补一条等价的新路径, 结点上的循环用闭包表示. 保持新路径与被删掉的路径集合等价.

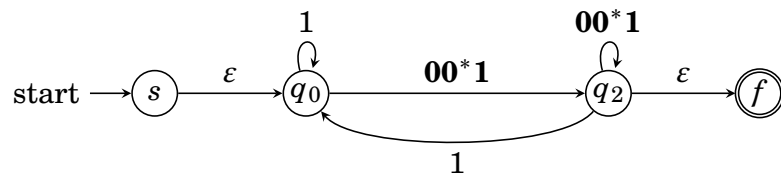
例 10. 利用状态消除法, 设计下图自动机的正则表达式.



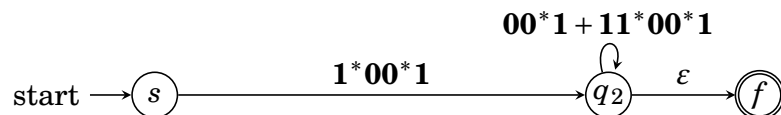
1. 利用空转移, 添加新的开始 s 和结束状态 f :



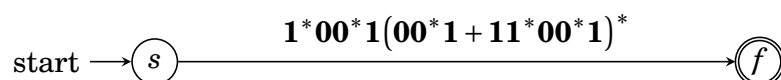
2. 消除状态 q_1 , 添加路径 $q_0 \rightarrow q_2$ 和 $q_2 \rightarrow q_0$:



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 ε -NFA

定理 4. 正则表达式定义的语言, 都可被有穷自动机识别.

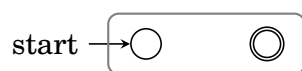
由正则表达式构造 ε -NFA(比定理 4 更严格的命题)

任何正则表达式 \mathbf{r} , 都存在等价的 ε -NFA A , 即 $L(A) = L(\mathbf{r})$, 并且 A 满足:

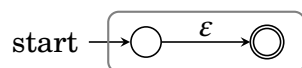
1. 仅有一个接收状态;
2. 没有进入开始状态的边;
3. 没有离开接受状态的边.

证明: 归纳基础:

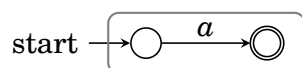
1. 对于 \emptyset , 有 ε -NFA:



2. 对于 ε , 有 ε -NFA:



3. $\forall a \in \Sigma$, 对于 \mathbf{a} , 有 ε -NFA:

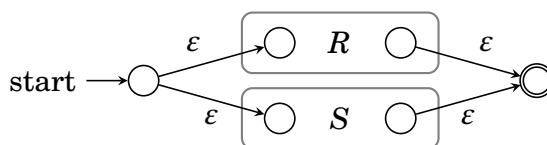


归纳递推: 假设正则表达式 \mathbf{r} 和 \mathbf{s} 的 ε -NFA 分别为 R 和 S



那么 $\mathbf{r+s}$, \mathbf{rs} 和 $\mathbf{r^*}$, 可由 R 和 S 分别构造如下:

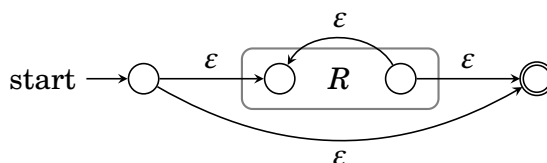
1. 对于 $\mathbf{r+s}$, 有 ε -NFA:



2. 对于 \mathbf{rs} , 有 ε -NFA:

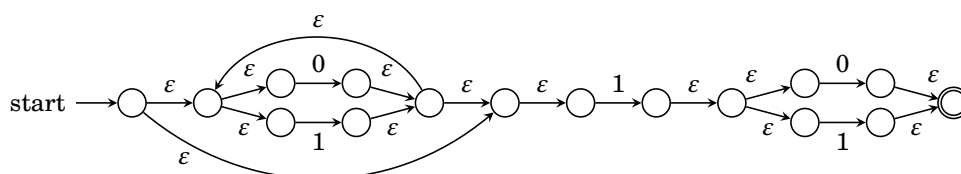


3. 对于 \mathbf{r}^* , 有 ε -NFA:



因此任何结构的正则表达式, 都有等价的 ε -NFA. □

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



思考题

正则表达式到 ε -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

$(L + M) + N = L + (M + N)$	结合律
$L + M = M + L$	交换律
$L + L = L$	幂等律
$\emptyset + L = L + \emptyset = L$	单位元 \emptyset

- 连接运算

$(LM)N = L(MN)$	结合律
-----------------	-----

$$\epsilon L = L \epsilon = L$$

单位元 ϵ

$$\emptyset L = L \emptyset = \emptyset$$

零元 \emptyset

$$LM \neq ML$$

- 分配率

$$L(M + N) = LM + LN$$

左分配律

$$(M + N)L = ML + NL$$

右分配律

- 闭包运算

$$(L^*)^* = L^*$$

$$\emptyset^* = \epsilon$$

$$\epsilon^* = \epsilon$$

$$L^* = L^+ + \epsilon$$

$$(\epsilon + L)^* = L^*$$

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 $\mathbf{a + ba} \stackrel{?}{=} (\mathbf{a + b})\mathbf{a}$;
3. 因为 $aa \notin \mathbf{a + ba}$ 而 $aa \in (\mathbf{a + b})\mathbf{a}$;
4. 所以 $\mathbf{a + ba} \neq (\mathbf{a + b})\mathbf{a}$;
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 $\mathbf{a, b, c}$ 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 $\mathbf{a^* = a^* a^*}$ 得 $L^* = L^* L^*$ 成立.

- $(L + M)^* M \stackrel{?}{=} (L^* M)^*$
替换得 $(\mathbf{a + b})^* \mathbf{b} \stackrel{?}{=} (\mathbf{a^* b})^*$, 因为 $\varepsilon \notin (\mathbf{a + b})^* b$ 且 $\varepsilon \in (\mathbf{a^* b})^*$, 所以不相等.
- $(R + S)^* \stackrel{?}{=} R^* + S^*$
- $(RS + R)^* R \stackrel{?}{=} R(SR + R)^*$
- $(RS + R)^* RS \stackrel{?}{=} (RR^* S)^*$
- $(R + S)^* S \stackrel{?}{=} (R^* S)^*$

3.4 练习题

1. Give regular expressions for each of the following languages over the alphabet $\{0, 1\}$.

(1) Strings that end with the suffix $0^5 = 00000$

- (2) All strings containing the substring 000.
 - (3) All strings *not* containing the substring 000.
 - (4) All strings *not* containing the substring 110.
 - (5) Every string except 000.
 - (6) All strings w such that in every prefix of w , the number of 0s and 1s differ by at most 1.
 - (7) All strings w such that in every prefix of w , the number of 0s and 1s differ by at most 2.
 - (8) All strings that start with 00 and contain at least one 1.
 - (9) All strings that start with 01 and contain at least two 0s.
 - (10) All strings that start with 1 and contain at least two 0s.
 - (11) All strings containing at least two 0s and at least one 1.
 - (12) All strings in which the substring 000 appears an even number of times. (For example, 0001000 and 10000 are in this language, but 000110000 and 00100000 are not.)
 - (13) Strings in which every occurrence of the substring 00 appears before every occurrence of the substring 11.
 - (14) Strings in which the number of 0s and the number of 1s differ by a multiple of 3.
 - (15) Strings that contain an even number of 1s and an odd number of 0s.
 - (16) Strings that represent a number divisible by 5 in binary.
2. [Exercise 3.1.1] Write regular expressions for the following languages:
- (a) The set of strings over alphabet $\{a, b, c\}$ containing at least one a and at least one b .
 - (b) The set of strings of 0's and 1's whose tenth symbol from the right end is 1.
 - (c) The set of strings of 0's and 1's with at most one pair of consecutive 1's.
3. [Exercise 3.1.2] Write regular expressions for the following languages:
- (a) The set of all strings of 0's and 1's such that every pair of adjacent 0's appears before any pair of adjacent 1's.
 - (b) The set of strings of 0's and 1's whose number of 0's is divisible by five.
4. [Exercise 3.1.3] Write regular expressions for the following languages:
- (a) The set of all strings of 0's and 1's not containing 101 as a substring.

- (b) The set of all strings with an equal number of 0's and 1's, such that no prefix has two more 0's than 1's, nor two more 1's than 0's.
- (c) The set of all strings of 0's and 1's whose number of 0's is divisible by five and whose number of 1's is even.
- (d) The set of all strings of 0's and 1's whose number of 0's is even and whose number of 1's is even.
5. [!Exercise 3.1.4] Give English descriptions of the languages of the following regular expressions:

(a) $(1 + \epsilon)(00^*1)^*0^*$

(b) $(0^*1^*)^*000(0+1)^*$

(c) $(0+10)^*1^*$

6. [!Exercise 3.1.4] \emptyset and $\{\epsilon\}$ are only two languages whose closure is finite.

7. [Exercise 3.2.1]: Here is a transition table for DFA:

	0	1
$\rightarrow q_1$	q_2	q_1
q_2	q_3	q_1
$*q_3$	q_3	q_2

- (a) Give all the regular expressions $R_{ij}^{(0)}$. *Note:* Think of state q_i as if it were the state with integer number i .
- (b) Give all the regular expressions $R_{ij}^{(1)}$. Try to simplify the expressions as much as possible.
- (c) Give all the regular expressions $R_{ij}^{(2)}$. Try to simplify the expressions as much as possible.
- (d) Give a regular expression for the language of the automaton.
- (e) Construct the transition diagram for the DFA and give a regular expressions for its language by eliminating state q_2 .
8. [Exercise 3.2.2]: Repeat Exercise 3.2.1 for the following DFA:

Note that solutions to parts (a), (b) and (e) are *not* available for this exercise.

	0	1
$\rightarrow q_1$	q_2	q_3
q_2	q_1	q_3
$*q_3$	q_2	q_1

9. [Exercise 3.2.3]: Convert the following DFA to a regular expression, using the state-elimination technique of Section 3.2.2.

	0	1
$\rightarrow^* p$	s	p
q	p	s
r	r	q
s	q	r

10. [Exercise 3.2.4]: Convert the following regular expressions to NFA's with ε -transitions.
- (a) 01^* .
 - (b) $(0+1)01$.
 - (c) $00(0+1)^*$.
11. [Exercise 3.2.5]: Eliminate ε -transitions from your ε -NFA's of Exercise 3.2.4. A solution to part (a) appears in the book's Web pages.

chunyu@hit.edu.cn

<http://iilab.net/chunyu>

