第三章　正则表达式 2025

本章内容：正则表达式RE

- 3.1 正则表达式（regular expression, RE）：一种类似于算术表达式的代数式子，作为匹配符号串的模式来定义语言。
  - 正则表达式是如何构成的？（定义）
  - 正则表达式如何定义语言？（语言识别器）
- 3.2 RE与DFA、NFA等价性
- 3.3 RE的典型应用
- 3.4 RE满足哪些代数定律？

构造正则语言

- 观察字母表Σ上的语言$L\subseteq\Sigma^*$：
  - **原子语言**：$\{a\}, a\in\Sigma$
  - **原子语言**：$\{\varepsilon\}$
  - **原子语言**：$\emptyset$
- 不能分解为由其他语言经过**运算**而成。
- 断言：原子语言都是正则语言。
  - 容易找到它们的FA。
- 试考虑这样一种**构造语言的思路**：
  - 通过原子语言与语言上的运算构造语言；
  - 同时断言如此构造的语言是正则语言。

语言的运算

- 令L, L_1, L_2 是Σ上的语言。

- 语言连接运算“·”借助于符号串的连接运算计算结果
  - $L_1 \cdot L_2 = \{s \cdot t \mid s \in L_1, t \in L_2\}$
  - “·”可省略
  - 满足结合律：$L(L_1L_2) = (LL_1)L_2$
  - n个L依次连接（n次幂）$L^n = \{s_1s_2...s_n \mid s_1, s_2, ..., s_n \in L\}$
  - $L^0 = \{\varepsilon\}$
- 语言上的闭包运算“*”
  - $L^* = L^0 \cup L^1 \cup L^2 \cup ... = \bigcup_{n \geqslant 0} L^n$
- 语言上的并运算“∪”就是集合并运算。

语言的连接运算、并运算

- $L_1 = \{0, 01\}$
- $L_2 = \{\varepsilon, 1, 11, 111, \ldots\}$
- $L_1L_2 = \{0, 01, 011, 0111, \ldots\} \cup \{01, 011, 0111, \ldots\}$
- $L_1{}^2 = \{00, 001, 010, 0101\}$
- $L_2{}^n = L_2\ (n \geqslant 1)$
- $L_1 \cup L_2 = \{0, 01, \varepsilon, 1, 11, 111, \ldots\}$

语言的闭包运算例

$L_1=\{0,01\},\ L_1{}^*=?$

$L_1{}^*$: $0|01|0|0|0|01$属于$L_1{}^*$

00110001不属于$L_1{}^*$

10010001不属于$L_1{}^*$

$L_1{}^0=\{\varepsilon\}$

$L_1{}^1=\{0,01\}$

$L_1{}^2=\{00,001,010,0101\}$

$L_1{}^3=\{000,0001,0010,00101,$
$0100,01001,01010,010101\}$

$L_1{}^*$所有以0开头不含连续1的串（加上空串）

语言闭包运算例

- 由任意个1组成的串的全集L_2

$$L_2 = \{\varepsilon, 1, 11, 111, \ldots\},\ \ L_2{}^{*} = ?$$

$$L_2{}^0 = \{\varepsilon\}$$
$$L_2{}^1 = L_2$$
$$L_2{}^2 = L_2$$
$$L_2{}^n = L_2 \quad (n \geqslant 1)$$

$$\begin{aligned} L_2{}^{*} &= L_2{}^0 \cup L_2{}^1 \cup L_2{}^2 \cup \ldots \\ &= \{\varepsilon\} \cup L_2{}^1 \cup L_2{}^2 \cup \ldots \\ &= L_2 \end{aligned}$$

$\varepsilon \in L^{*}$?	$\varepsilon \in L^{*}$
$\varnothing^{*} = ?$	$\varnothing^{*} = \{\varepsilon\}$
$\{\varepsilon\}^{*} = ?$	$\{\varepsilon\}^{*} = \{\varepsilon\}$
$\{1\}^{*} = L_2$?	$\{1\}^{*} = L_2$

构造语言的方法

- **原子语言**：Σ上最基本的语言$\varnothing$、$\{\varepsilon\}$、$\{a\}, a\in\Sigma$。
- **语言上运算**：**连接**、**闭包**、**并**。
- **构造方法**：从原子语言开始，利用语言上的运算逐步组合而成，其中可使用括号。
- 性质：
  - 原子语言是Σ上语言。
  - 如果L, L_1是Σ上语言，那么$L\cdot L_1$、$L\cup L_1$、L^*都是。
  - 任何Σ上语言的运算结果仍然是Σ上语言。
- **语言构造过程就是语言运算表达式的计算过程**。例子：
  - $L_1=\{0\}(\{0\}\cup\{1\})^*$
  - $L_2=(\{0\}\{1\}^*)\cup(\{1\}\{0\}^*)$
- 语言运算表达式模式化为正则表达式。

什么是正则表达式

- 原子语言的模式
  - 语言$\{a\}$模式化为RE a，有$L(a)=\{a\}$
  - 语言$\{\varepsilon\}$模式化为RE ε，有$L(\varepsilon)=\{\varepsilon\}$
  - $\varnothing$指代{}，即$L(\varnothing)=\emptyset$
- 运算符∪改名为+

RE运算符	语言运算符
连接'·'	连接'·'
并'+'	并'∪'
闭包'*'	闭包'*'

- 例：语言L_1是0或由0开头的0-1串组成
  - 语言运算表达式：　$\{0\}(\{0\}\cup\{1\})^*$
  - 对应正则表达式：　$0(0+1)^*$
  - $L(0(0+1)^*)=\{0\}(\{0\}\cup\{1\})^*=L_1$
- 例：$L_2=(\{0\}\{1\}^*)\cup(\{1\}\{0\}^*)$
  - 对应正则表达式：　01^*+10^*

构造正则表达式

- 字母表Σ上的正则表达式是使用如下规则形成的表达式。
- **基础：**
  - 符号 $\varnothing$ 和 ε 都是正则表达式；
  - Σ中每个元素 a 都是正则表达式；
- **归纳：**
  - 如果 R 和 S 是正则表达式，那么 $R+S$、RS 和 R^* 都是；
  - 如果R是正则表达式，那么(R)也是。

a) $0(0+1)^*$
b) 01^*+10^*
c) $(0+1)^*01(0+1)^*$

- 正则表达式运算符的优先级
  - 优先级：'*'最高，'·'次之，'+'最低
  - 计算顺序：先括号里，后括号外，同级从左往右

归纳：字母表Σ上RE:r与$L(r)$间的一一对应

- R，S，T为正则表达式，$L(R)$为R定义的语言

正则表达式r	正则语言$L(r)$
$\varnothing$	$L(\varnothing)=\emptyset$
ε	$L(\varepsilon)=\{\varepsilon\}$
a	$L(a)=\{a\}$
R	$L(R)$
$R\cdot S$	$L(R\cdot S)=L(R)\cdot L(S)$
$R+S$	$L(R+S)=L(R)\cup L(S)$
R^*	$L(R^*)=L(R)^*$
$R+S+T=(R+S)+T$	$L(R+S+T)=L(R+S)\cup L(T)$
$R\cdot S\cdot T=(R\cdot S)\cdot T$	$L(R\cdot S\cdot T)=L(R\cdot S)\cdot L(T)$
$R+S\cdot T=R+(S\cdot T)$	$L(R+S\cdot T)=L(R)\cup L(S\cdot T)$
$R\cdot S^*=R\cdot(S^*)$	$L(R\cdot S^*)=L(R)\cdot L(S^*)$
$R+S^*=R+(S^*)$	$L(R)\cup L(S^*)$

例：分析正则表达式

- 以下例子都是关于RE的语言

$0+1$ 长度为1的0-1串$=\{0, 1\}$

$(0+1)^*$ 任意0-1串$\{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots\}$

$L((0+1)^*)=\{0,1\}^*$

$(0+1)^*010$ 以010结尾的0-1串

$(0+1)^*11(0+1)^*$ 包含子串11的0-1串

例：分析正则表达式

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

长度是偶数或是3的倍数的0-1串

$((0+1)(0+1))^*$ 偶数长度

$(0+1)(0+1)$ 长度为2

$((0+1)(0+1)(0+1))^*$ 长度3的倍数

$(0+1)(0+1)(0+1)$ 长度为3

例：分析正则表达式

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

串可以分成段，
其中每段长度为2或3

$$(0+1)(0+1)+(0+1)(0+1)(0+1)$$

串长度为2或 3

$$(0+1)(0+1)$$

长度2

$$(0+1)(0+1)(0+1)$$

长度3

例：分析正则表达式

$$(1+01+001)^*(\varepsilon+0+00)$$

以最多两个0结束

连续的1之间被最多两个0分开

不会有连续三个0的串

$$L((1+01+001)^*(\varepsilon+0+00)) = \{x: x \text{ 不含子串} 000\}$$

ε　　00　　01|1|001|01|1|0　　001|001|0

例：写出正则表达式

- 该语言元素包含子串00
  - $(0+1)^*00(0+1)^*$
- 该语言元素不含子串00
  - $1^*(011^*)^*(\varepsilon+0)$

$$\underbrace{111}_{\text{开头有些1}}\underbrace{011\,|\,01\,|\,011\,|\,01\,|\,01}_{\text{每个0后跟1到多个1}}\underbrace{0}_{\text{有可能以一个0为结束}}$$

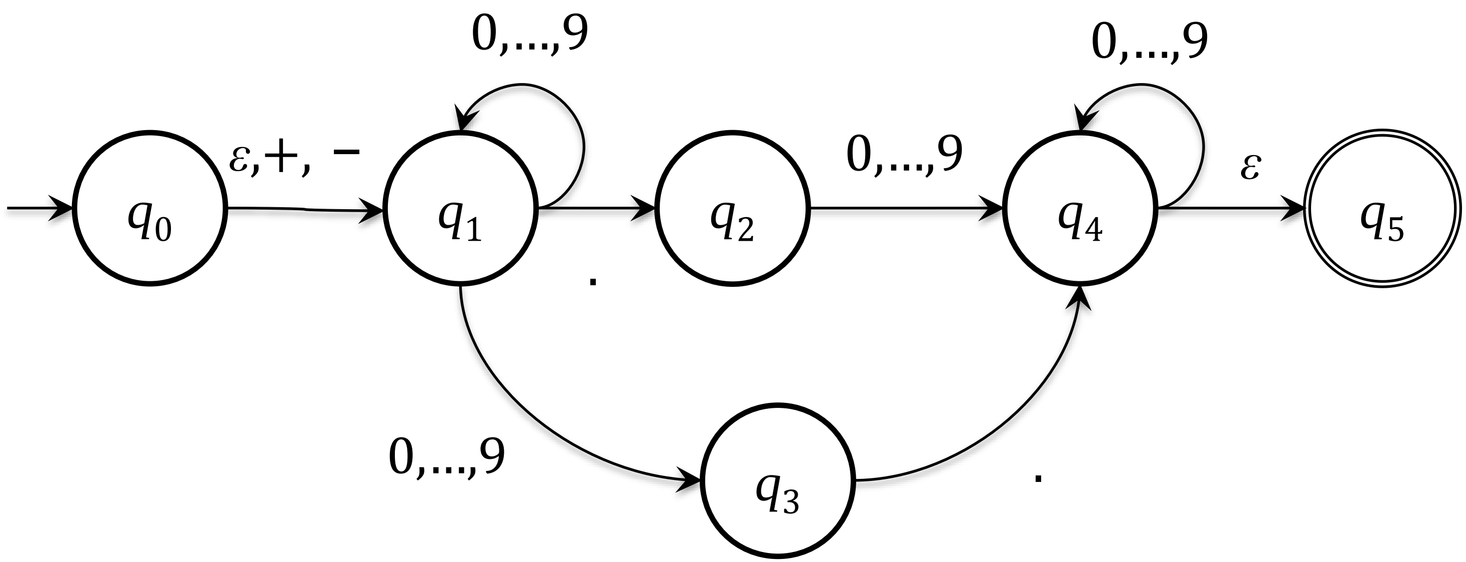
  - 可有全1前缀，也可没有
  - 中间部分，每个0后跟1到多个1
  - 可有0为后缀，也可没有

例：写出正则表达式

- 该语言元素都有偶数个0
  - $(1^*01^*01^*)^*$
- 该语言元素有偶数个0，或者有奇数个1
  - $(1^*01^*01^*)^*+(0^*10^*)(0^*10^*10^*)^*$
- 该语言元素有偶数个0，并且有奇数个1
  - ?

例：写出正则表达式

- 有符号十进制定点数。



- $(\varepsilon+\backslash++-)((0+\ldots+9)^{*}.(0+\ldots+9)(0+\ldots+9)^{*}+(0+\ldots+9)(0+\ldots+9)^{*}.(0+\ldots+9)^{*})$

3.2 RE与DFA、NFA等价性

- 一个语言用正则表达式定义、也用有穷自动机定义，如何？

- **定理**3.1: **对任一**DFA A **存在正则表达式** R **使得** $L(R)=L(A)$
- **定理**3.2: **对任意正则表达式** R **存在** ε-NFA E **使得** $L(E)=L(R)$

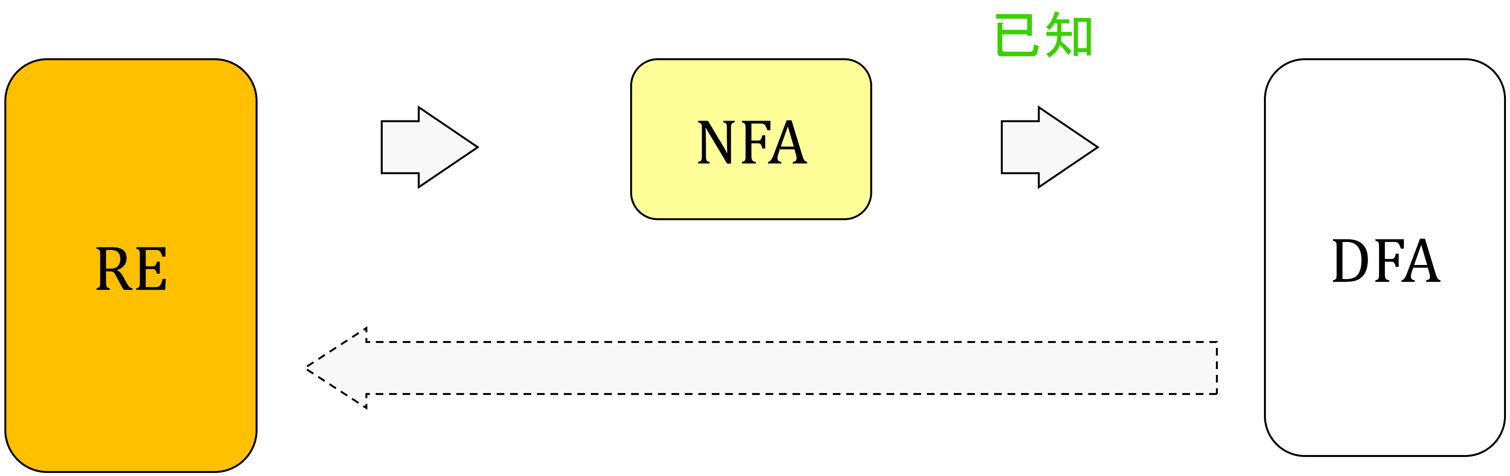
DFA ⟺ NFA

②定理3.1 ①定理3.2

RE

Kleene 定理

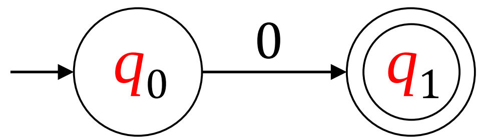
定理3.2证明路线图



每一个用正则表达式来定义的语言也可用有穷自动机来定义

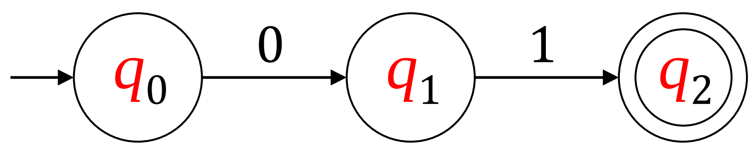
例：从正则表达式到 NFA

$R_1 = 0$



原子语言

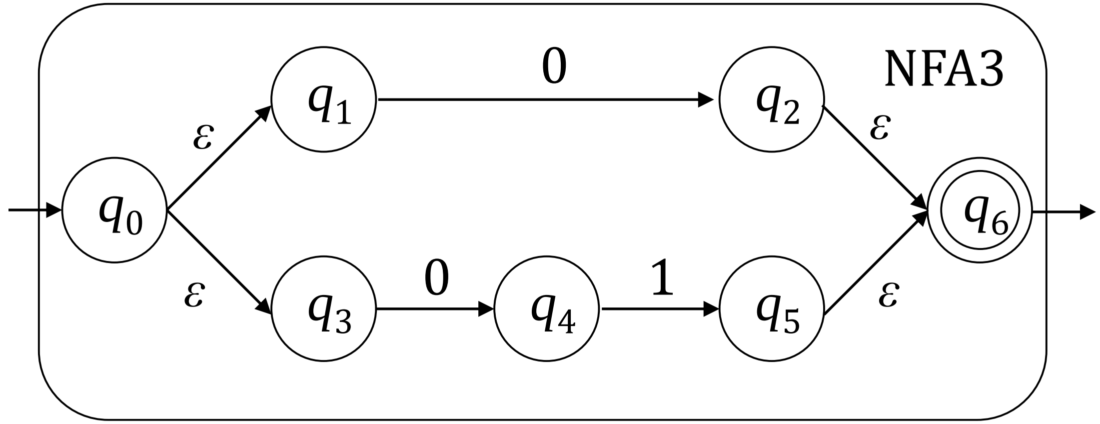
$R_2 = 01$



连接运算

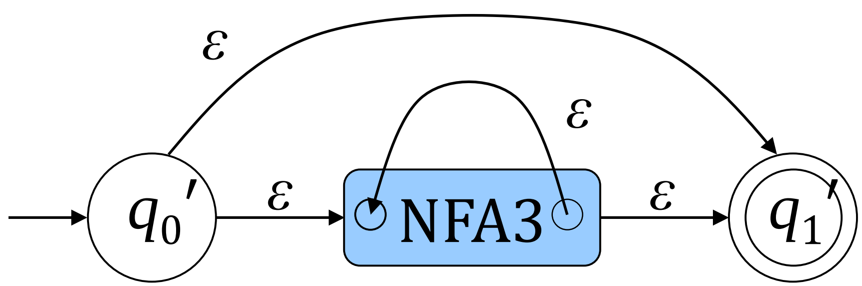
例：从正则表达式到 NFA

$R_3 = 0 + 01$



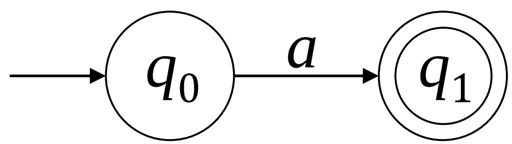
并

$R_4 = (0 + 01)^*$



闭包

正则表达式转换为NFA通用方法

正则表达式	NFA
$\varnothing$	
ε	
$a \in \Sigma$	

基础:

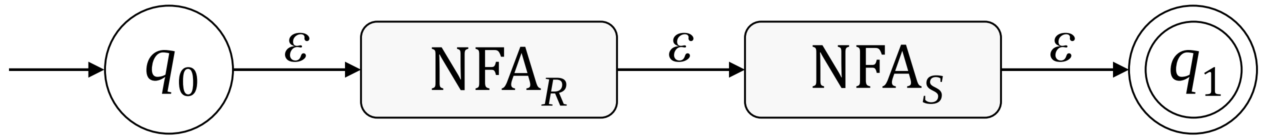
- 正则表达式$\varnothing$，ε和a, $a \in \Sigma$对应的NFA如图所示。

归纳

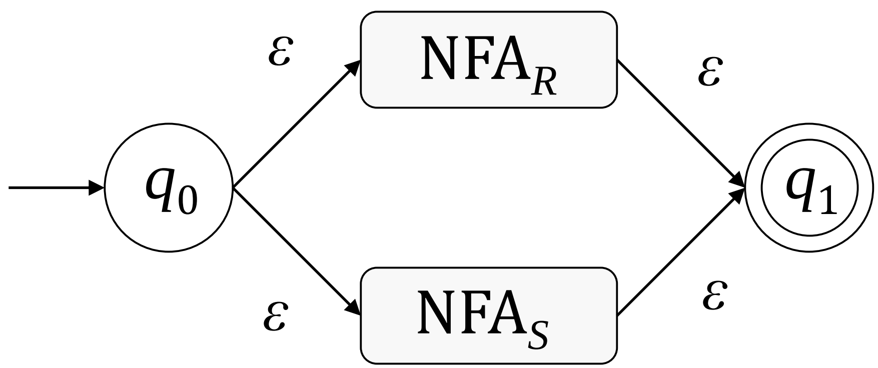
- 对于正则表达式R和S，$R+S$, RS, R^*和(R)，？

归纳证明示意

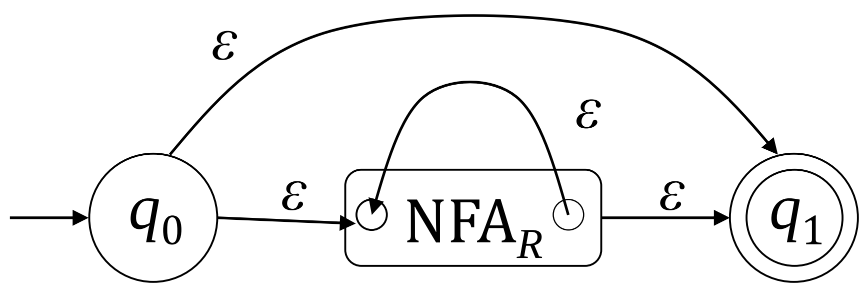
RS



$R+S$

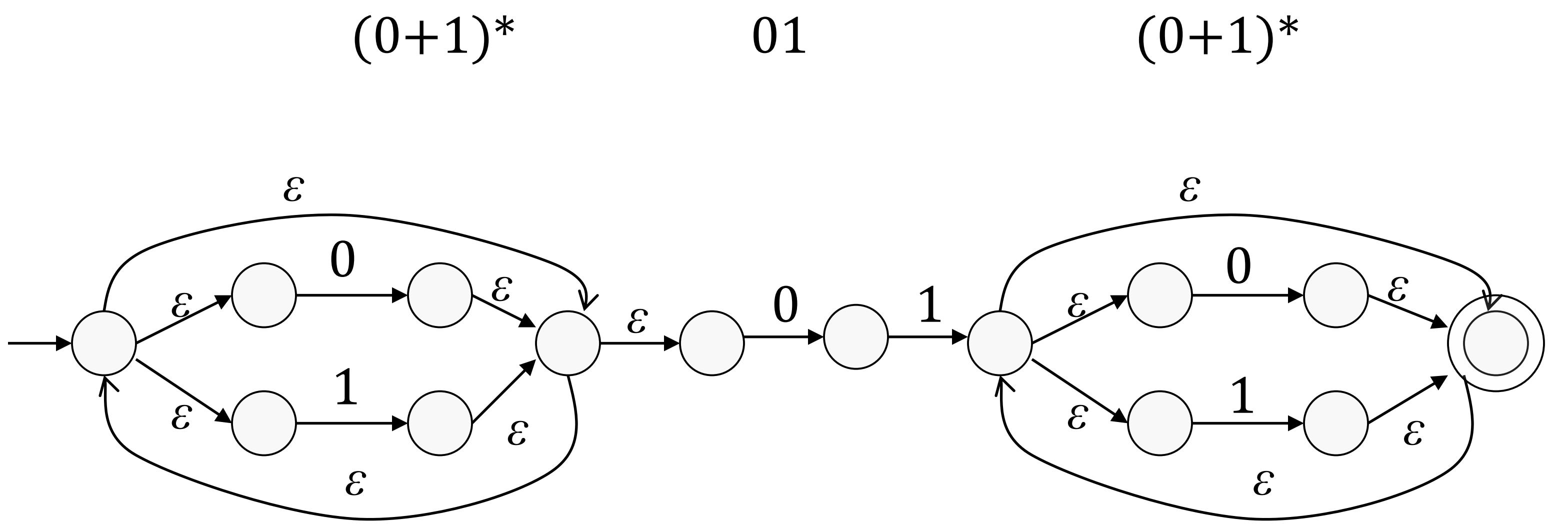


R^*



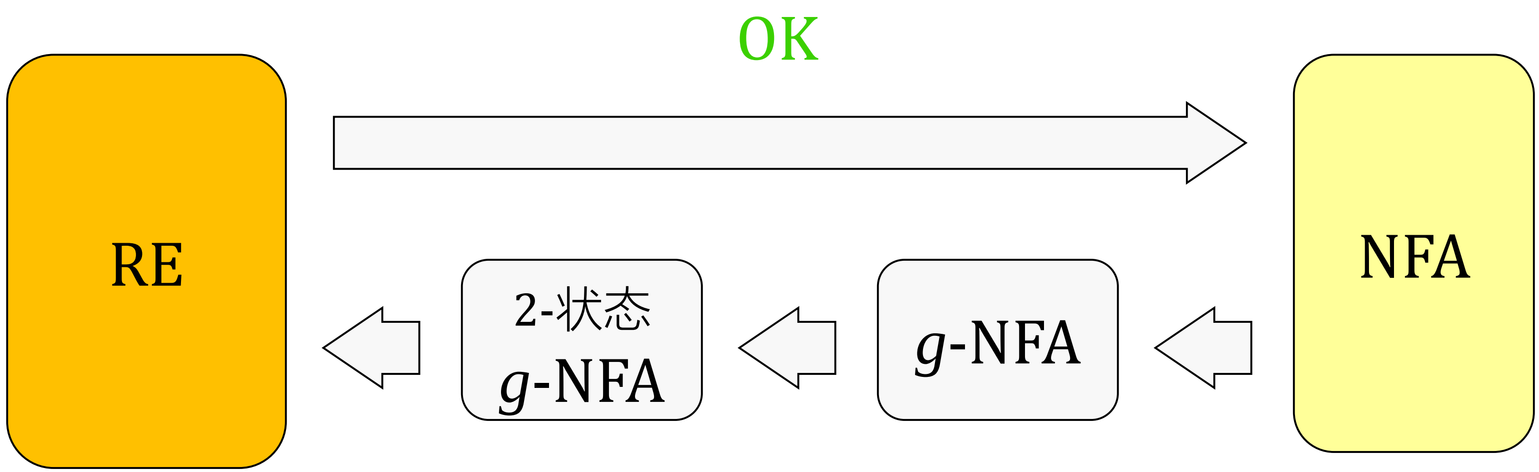
例：从RE构造NFA

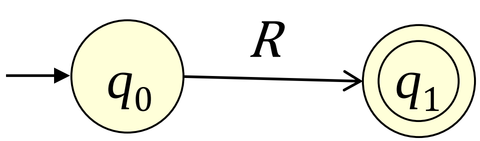
- (0+1)*01(0+1)*



有穷自动机转换为RE

- 对任一DFA A 存在正则表达式 R 使得 $L(R)=L(A)$
- DFA是NFA特例，所以关键是NFA到RE转换方法

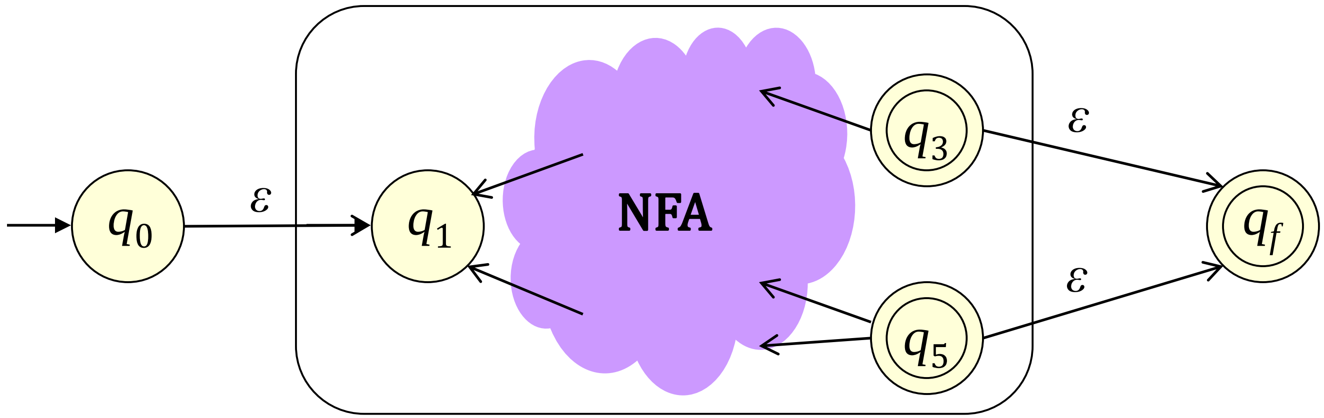




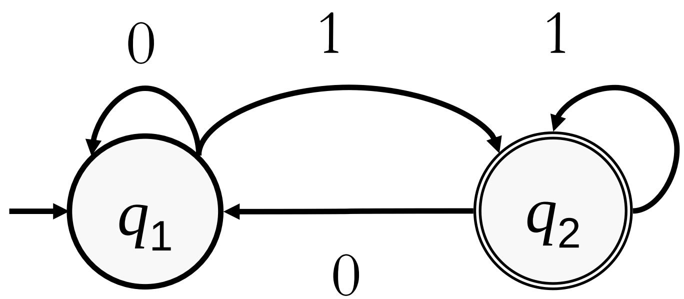
任一个 2-状态 g-NFA 总与某个正则表达式 R 等价。

通用NFA（g-NFA）

- 只有一个接受状态
- 没有射入到初始状态的弧
- 没有从接受状态射出的弧
- 弧上标记采用正则表达式作标记

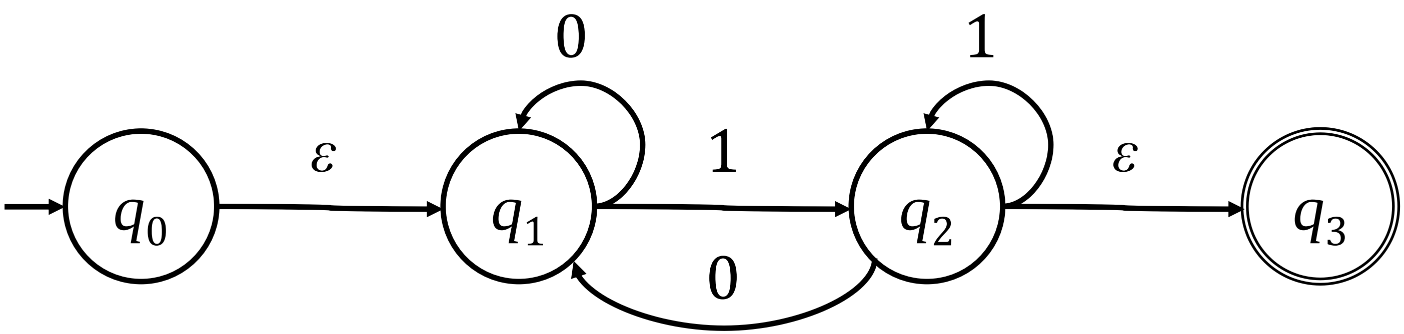


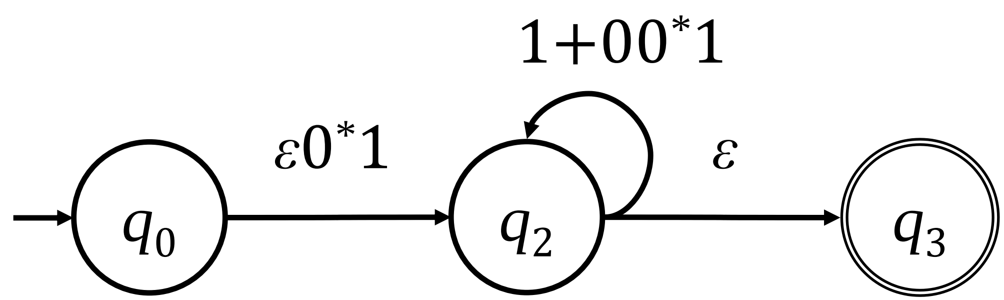
g-NFA



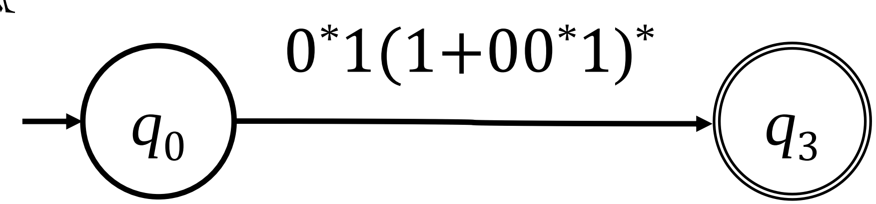
- 只有一个接受状态？YES
- 没有射入到初始状态的弧？NO
- 没有从接受状态射出的弧？NO
- 弧上标记是正则表达式？NO/YES

g-NFA



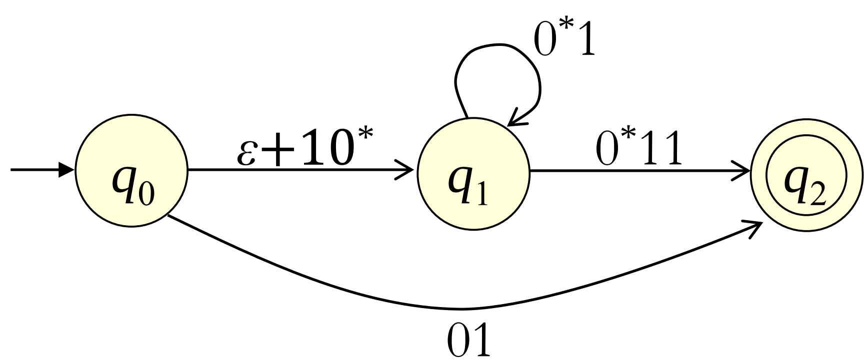


- √ – 只有一个接受状态
- √ – 没有射入到初始状态的弧
- √ – 没有从接受状态射出的弧

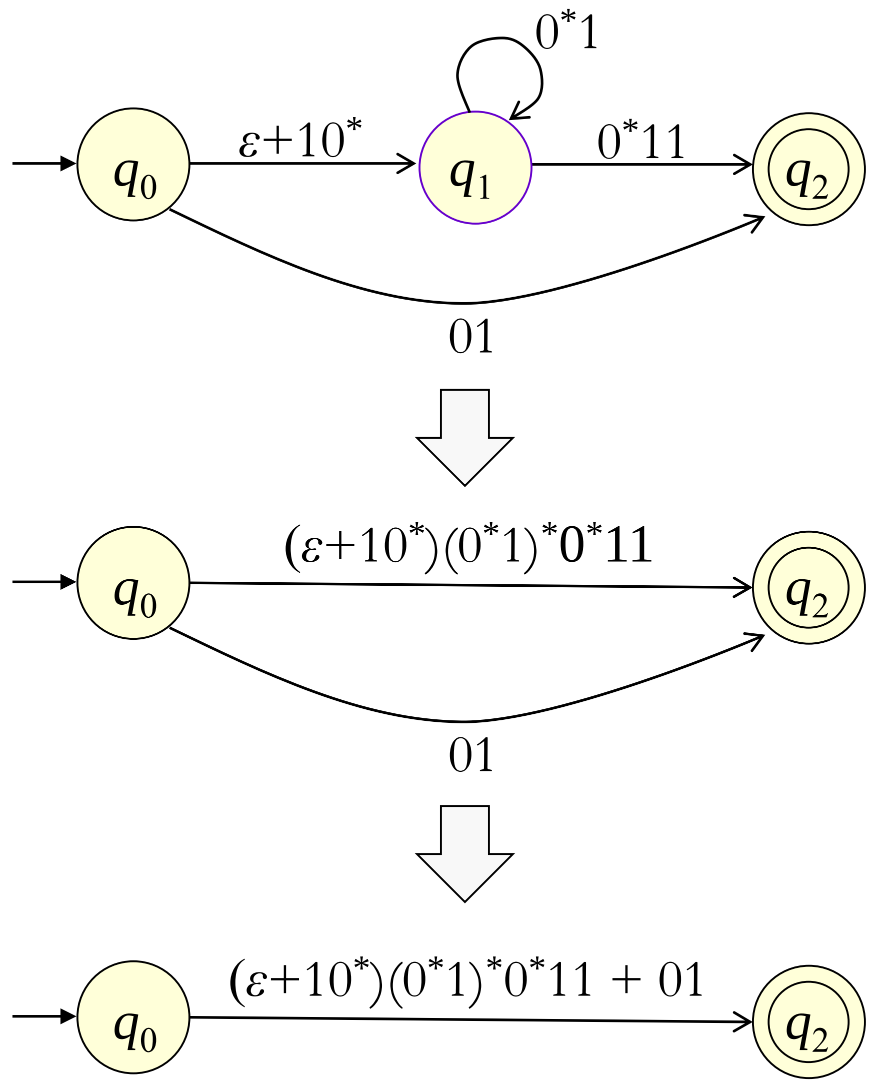


NFA与g-NFA对比

- g-NFA弧上标记采用正则表达式作标记
- NFA的a弧看作正则表达式a做标记
- NFA的ε弧看作正则表达式ε做标记
- NFA的缺失弧可看作是正则表达式$\varnothing$做标记的弧
- 如此，NFA弧上标记都被当作正则表达式，再检查没有射入初始状态的弧和没有从接受状态射出的弧，即成为g-NFA



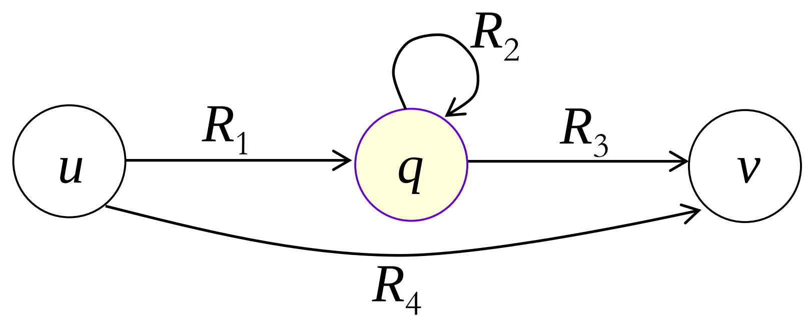
g-NFA的状态约简


q_0
q_1
q_2
$\varepsilon+10^*$
0^*1
0^*11
01
$(\varepsilon+10^*)(0^*1)^*\mathbf{0}^*11$
01
$(\varepsilon+10^*)(0^*1)^*0^*11 + 01$

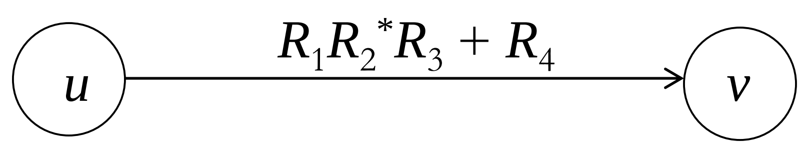
g-NFA状态约简——通用规则

- 要消除状态q, 对于每个状态对 (u, v)

替代



为



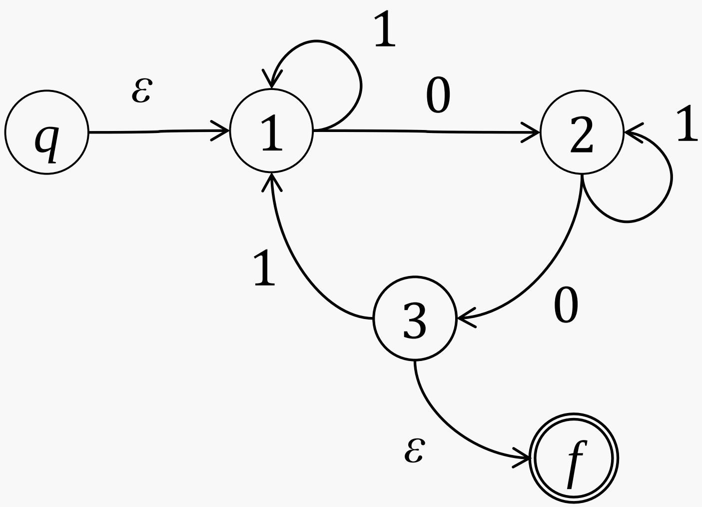
- 注意u=v时也如此

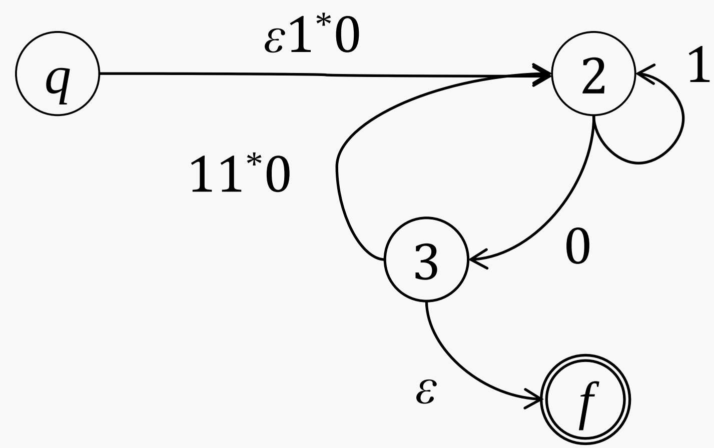
对于要消除的节点s进行不无遗漏的处理：

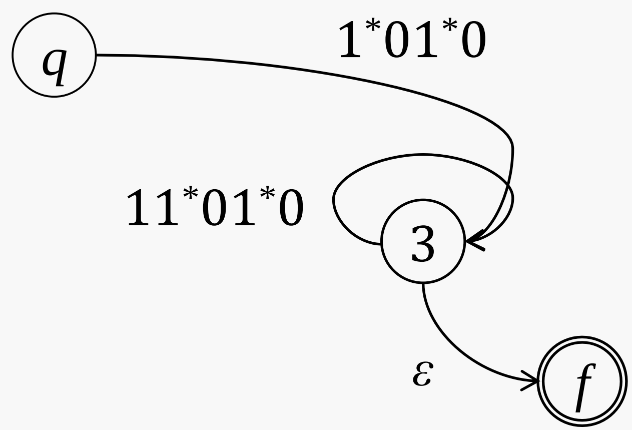
$\lambda(s)$头为s的弧尾的集合，自环除外，比如u

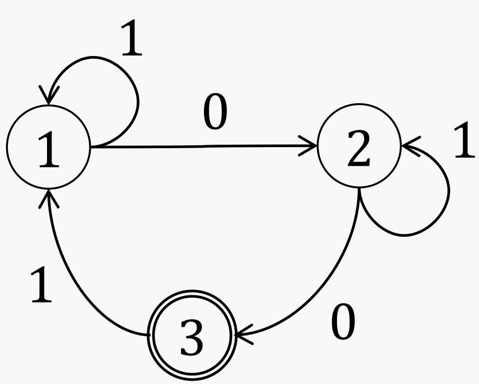
$\mu(s)$尾为s的弧头的集合，自环除外，比如v

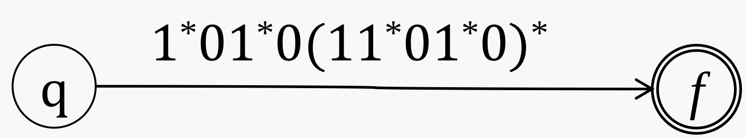
例： *g*-NFA状态约简


q
ε
1
1
0
2
1
1
3
0
ε
f


q
ε1*0
2
1
11*0
3
0
ε
f


q
1*01*0
11*01*0
3
ε
f


1
1
0
2
1
1
3
0


1*01*0(11*01*0)*
q
f

定理3.1证明

- 定理3.1 对于任意一个DFA A，存在一个正则表达式R，使得$L(A)=L(R)$。
- DFA $A=(Q,\Sigma,\upsilon,q_0,F)$
- $L(A)=\{w \mid w\in\Sigma^*, \tilde{\upsilon}(q_0,w)\in F\}$
- 证明的思路：
  - 考察q_0至q，$q\in F$，之间所有路径，用正则表达式表示之，即得。
  - 这些路径是否不无遗漏地都表示为正则表达式了呢？

关键思路：用归纳法穷尽所有路径

- 对结点编号，对两端点间所有路径进行排序以便于归纳。
- 对DFA状态图中的顶点从1到n，$n=|Q|$，编号。
- 对始端i到末端j的所有路径，$1\leqslant i,j\leqslant n$，进行如下排列：
  - 路径上除端点外的所有中间结点其编号不大于k，按照$k=0,1,\ldots,n$依次对路径进行排列
  - 注意$k=0$时满足条件的路径为只有端点，没有中间结点
- 那么针对路径PATH(i,j)的归纳方式：
  - 基础：不经过任何中间结点的路径的标记为$R_{ij}^{(0)}$
  - 归纳：经过不大于k的中间结点的路径的标记为

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

证明定理3.1

- 基础：不经过任何中间结点的路径为$R_{ij}^{(0)}$，分两种情形：
- 情形一：$i \neq j$;
  - $R_{ij}^{(0)} = \varnothing$;
  - for $(a \in \Sigma)$ if $(j \in \upsilon(i, a))$ $R_{ij}^{(0)} = R_{ij}^{(0)} + a$;
- 情形二：$i = j$;
  - $R_{ij}^{(0)} = \varepsilon$;
  - for $(a \in \Sigma)$ if $(j \in \upsilon(i, a))$ $R_{\mathrm{ij}}^{(0)} = R_{\mathrm{ij}}^{(0)} + a$;
- 归纳：

证明定理3.1

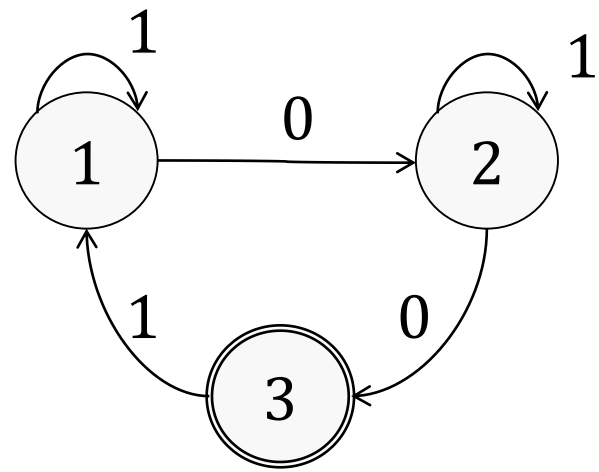
- 归纳：假定顶点i到顶点j所有经过顶点不大于$k-1$的路径，它们的标记并在一起定义为$R_{ij}^{(k-1)}$，那么i到j所有经过顶点不大于k的路径，分为两种情形：
- 情形一：没有经过顶点k；
  - 根据归纳假设，这些路径的标记都属于$L(R_{ij}^{(k-1)})$
- 情形二：经过顶点k；
  - 路径分段为：$\mathrm{PATH}(i,k)$，$\mathrm{PATH}(k,k)$，$\mathrm{PATH}(k,j)$，且各段都属于情形一
  - 根据归纳假设，这三段组合成的所有路径，它们的标记构成的集合定义为$R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^*R_{kj}^{(k-1)}$
- 最后，将两种情形合并在一起得到
  - $R_{ij}^{(k)}=R_{ij}^{(k-1)}+R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^*R_{kj}^{(k-1)}$

证明定理3.4

- 令$n=|Q|$，Q中元素编号为从1到n，其中q_0编号为1，
- 令$r=+j\in\{q$的编号$\mid q\in F\}\cdot R_{1j}^{(n)}$
  - 即$r=R_{1j1}^{(n)}+\ldots+R_{1jm}^{(n)}$，其中$j_1,\ldots,j_m$是$F$中各元素的编号
- 则$L(A)=L(r)$
- 证毕。

例：构造DFA的正则表达式

- $R_{ij}{}^{(k)} = R_{ij}{}^{(k-1)} + R_{ik}{}^{(k-1)}(R_{kk}{}^{(k-1)})^{*}R_{kj}{}^{(k-1)}$
- $r = R_{13}{}^{(3)} = R_{13}{}^{(2)} + R_{13}{}^{(2)}(R_{33}{}^{(2)})^{*}R_{33}{}^{(2)}$



$R_{13}{}^{(2)} = R_{13}{}^{(1)} + R_{12}{}^{(1)}(R_{22}{}^{(1)})^{*} R_{23}{}^{(1)}$

$R_{13}{}^{(1)} = \varnothing$

$R_{12}{}^{(1)} = R_{12}{}^{(0)} + R_{11}{}^{(0)} (R_{11}{}^{(0)})^{*}R_{12}{}^{(0)})$

$R_{22}{}^{(1)} = R_{22}{}^{(0)}$

$R_{23}{}^{(1)} = R_{23}{}^{(0)}$

$R_{33}{}^{(2)} = R_{33}{}^{(1)} + R_{32}{}^{(1)}(R_{22}{}^{(1)})^{*} R_{23}{}^{(1)}$

$R_{33}{}^{(1)} = R_{33}{}^{(0)} + R_{31}{}^{(0)}(R_{11}{}^{(0)})^{*} R_{13}{}^{(0)}$

$R_{32}{}^{(1)} = R_{32}{}^{(0)} + R_{31}{}^{(0)}(R_{11}{}^{(0)})^{*} R_{12}{}^{(0)}$

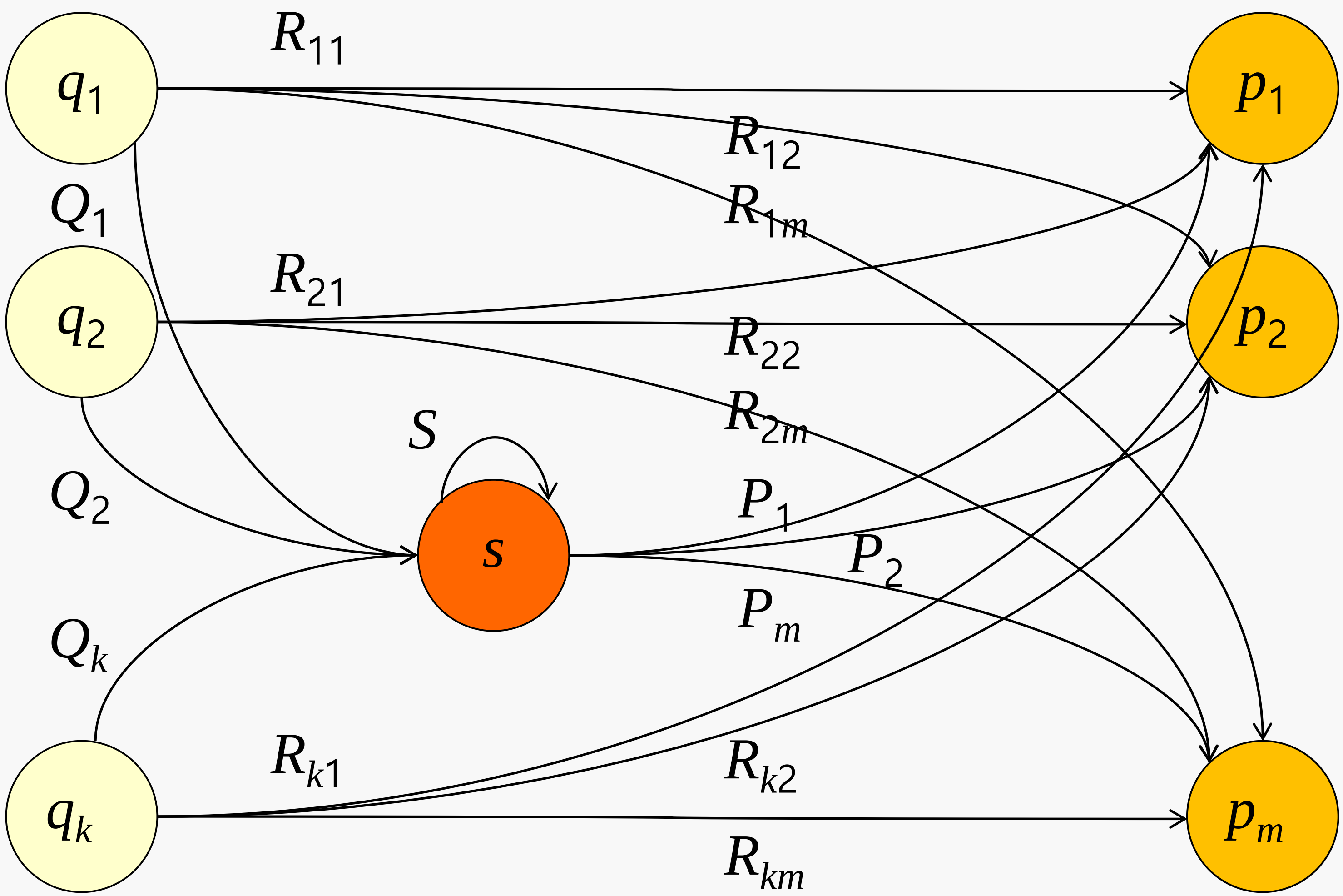
$R_{11}{}^{(0)}$	$\varepsilon+1$
$R_{12}{}^{(0)}$	0
$R_{22}{}^{(0)}$	$\varepsilon+1$
$R_{23}{}^{(0)}$	0
$R_{33}{}^{(0)}$	ε
$R_{13}{}^{(0)}$	$\varnothing$
$R_{31}{}^{(0)}$	1
$R_{32}{}^{(0)}$	$\varnothing$

$\varnothing + (0+(\varepsilon+1)^{*}0)(\varepsilon+1+(\varepsilon+1)^{*})\ (0) + (R_{33}{}^{(2)})^{*}$

$= (0+1^{*}0)\ (\varepsilon+1^{*})\ (0) + (R_{33}{}^{(2)})^{*}$

状态约简规则（消去s状态前）

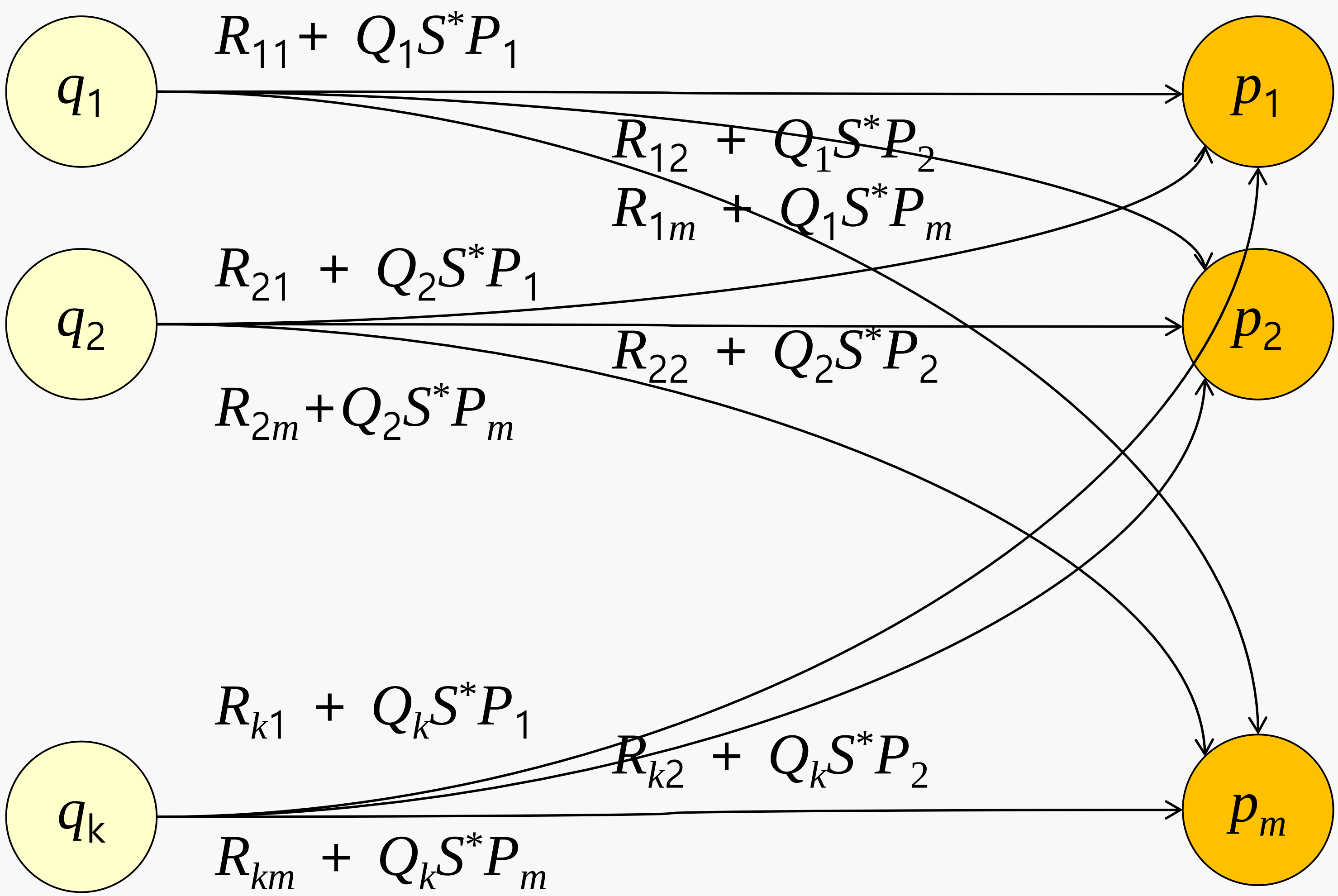
$\lambda(s)=\{q_1, q_2, \ldots, q_k\}$; $\mu(s)=\{p_1, p_2, \ldots, p_m\}$



对于$q\in\lambda(s)$, $p\in\mu(s)$, **有**R_{qp}, R_{qs}, R_{ss}, R_{sp}

状态约简规则（消除s状态后）

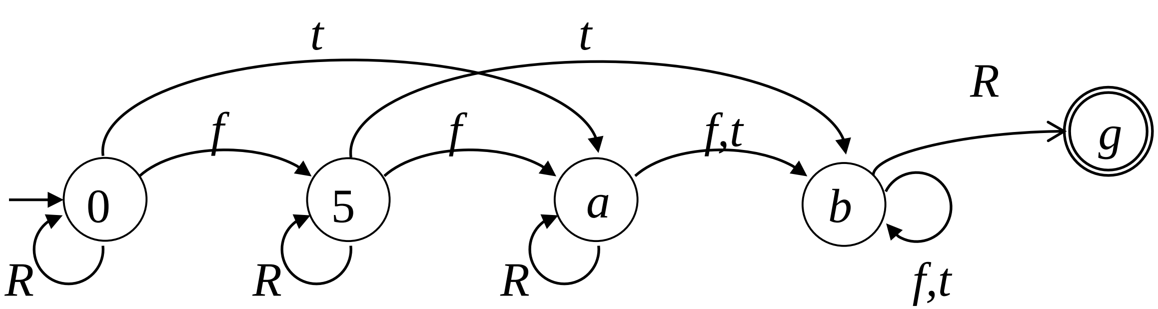
$$\forall q\in\lambda(s),\ p\in\mu(s)\cdot R_{qp}+R_{qs}R_{ss}{}^{*}R_{sp}$$



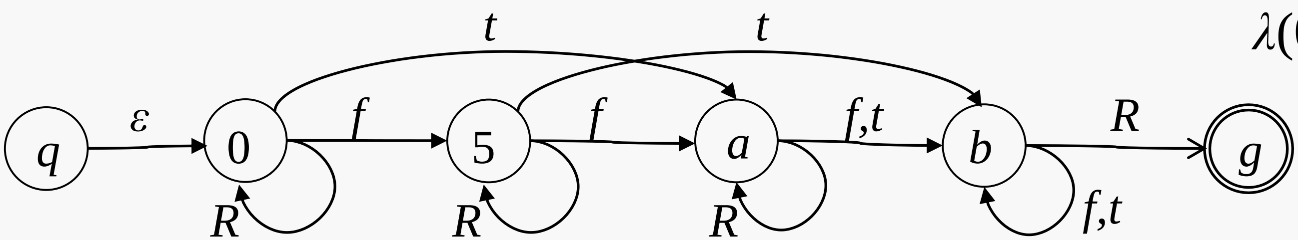
口香糖球机的语言

- 始端0末端g的路径有三类：
  - 0,5,a,b,g　　$R^*fR^*fR^*(f+t)(f+t)^*R$
  - 0,a,b,g　　$R^*tR^*(f+t)(f+t)^*R$
  - 0,5,b,g　　$R^*fR^*t(f+t)^*R$

$$R^*fR^*fR^*(f+t)(f+t)^*R + R^*tR^*(f+t)(f+t)^*R + R^*fR^*t(f+t)^*R$$



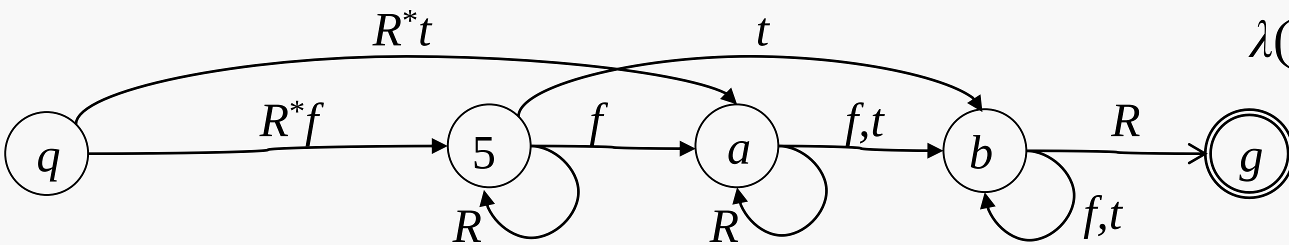
求口香糖球机的RE



$\lambda(0)=\{q\};\ \mu(0)=\{5, a\}$

$R_{q5}+R_{q0}R_{00}^{*}R_{05}$

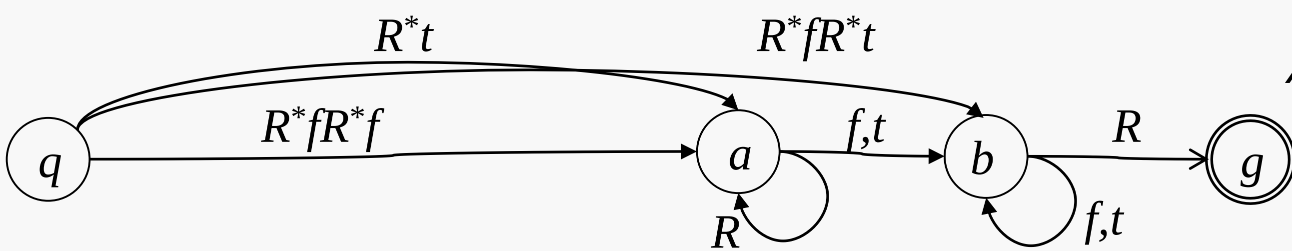
$R_{qa}+R_{q0}R_{00}^{*}R_{0a}$



$\lambda(5)=\{q\};\ \mu(5)=\{a, b\}$

$R_{qa}+R_{q5}R_{55}^{*}R_{5a}$

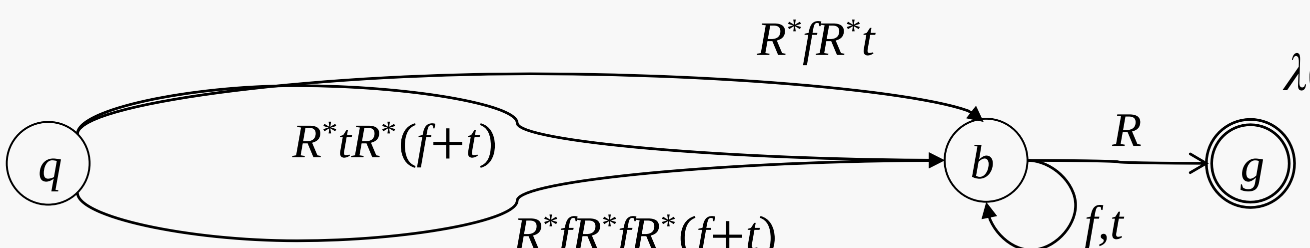
$R_{qb}+R_{q5}R_{55}^{*}R_{5b}$



$\lambda(a)=\{q\};\ \mu(a)=\{b\}$

$R_{qb}+R_{qa}R_{aa}^{*}R_{ab}$

$R_{qa}=?$

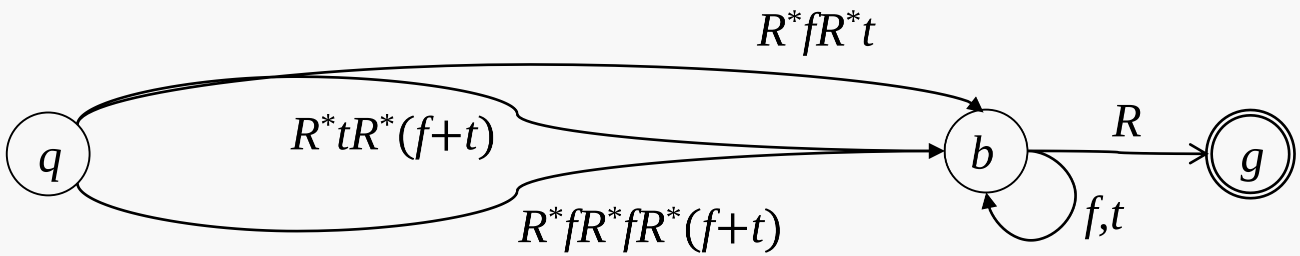


$\lambda(b)=\{q\};\ \mu(b)=\{g\}$

$R_{qg}+R_{qb}R_{bb}^{*}R_{bg}$

$R_{qb}=?$

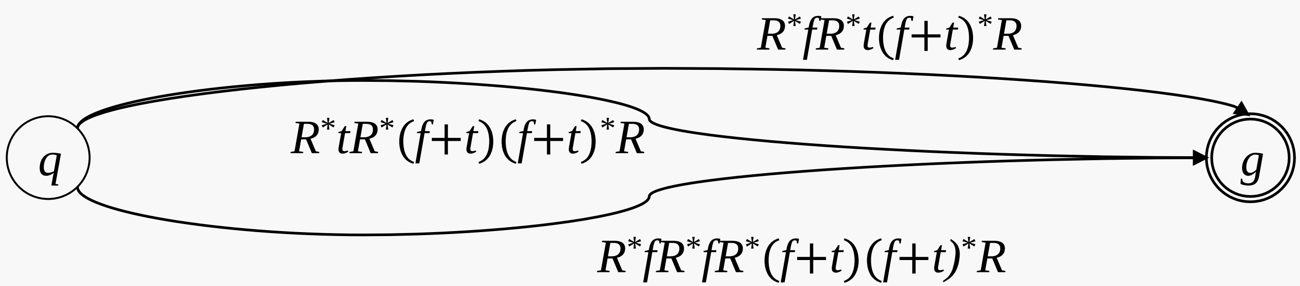
求口香糖球机的RE



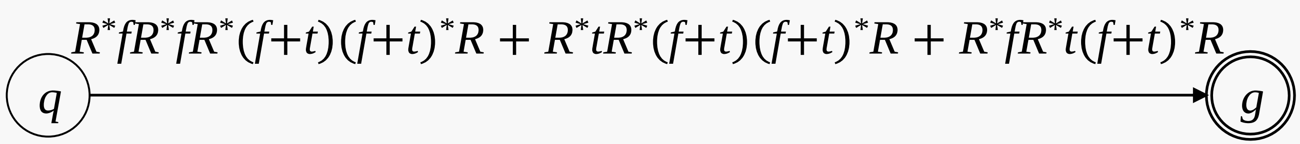
$\lambda(b)=\{q\};\ \mu(b)=\{g\}$

$R_{qg}+R_{qb}R_{bb}{}^{*}R_{bg}$

$R_{qb}=?$



$R_{qg}=?$



3.3 正则表达式应用举例

- 正则表达式是基于模式匹配定义了实际应用中感兴趣的符号串集合，可用于生成、检索、识别有用的文本内容。通常地，这种作为模式的正则表达式被自动构建成DFA以方便程序实现来完成上述功能。

- UNIX中的正则表达式扩展了RE的操作符
- 允许以RE为输入自动生成词法分析器
- 识别有用的文本内容

UNIX中的正则表达式

- 字母表较大时如ASCII字符集需要更为简洁的表达方式
- 元符号的概念，如元符号‘.’表示任意字符，那么‘\.’才是点
- 序列$[a_1a_2…a_k]$表示$a_1+a_2+…+a_k$
- 默认序列的子序列：[0-9]、[a-z]、[A-Za-z0-9]
- 增加的运算符：R?指R出现或不出现；R+指R重复1到多次；R{n}指恰有连续n个R；另外|取代+
- 整数：[+−]?[1-9][0-9]*|0
- 定点数：<整数>\.?|(<整数>|[+−]?)\.[0-9]*[1-9]
- 浮点数：<定点数>[e|E]<整数>

以RE为输入自动生成词法分析器

- 工具lex、flex
- 以正则表达式作为所识别的单词的模式提供给工具
- 工具将生成一个词法分析器，它按照这些模式一个一个识别输入流中的单词并返回结果
- 具体将在后面课程中介绍

识别有用的文本内容

- 如识别街道地址
- Street|St\.|Avenue|Ave\.|Road|Rd\.
- 有些街道包含多个单词如华盛顿特区的Rhode Island Avenue（罗德岛大道）、Xianning West Rd.
- ([A-Z][a-z]*\s)+(Street|St\.|Avenue|Ave\.|Road|Rd\.)
- \s匹配任何空白字符
- 加上门牌号码，如123A Main St
- [0-9]+[A-Z]?\s ([A-Z][a-z]*\s)+ (Street|St\.|Avenue|Ave\.|Road|Rd\.)
- 其它地址情形往上加…
- 28 XIANNING West Rd.
- 咸宁西路28号

识别有用的文本内容

- 正则表达式通常被用来检索、替换那些符合某个模式（规则）的文本。

- 用户名
- 电子邮箱
- IP地址

常用正则表达式

<table>
<tr><td>用户名</td><td>/^[a-z0-9_-]{3,16}$/</td></tr>
<tr><td>密码</td><td>/^[a-z0-9_-]{6,18}$/</td></tr>
<tr><td>十六进制值</td><td>/^#?([a-f0-9]{6}|[a-f0-9]{3})$/</td></tr>
<tr><td>电子邮箱</td><td>/^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/
/^[a-z\d]+(\.[a-z\d]+)*@([\da-z](-[\da-z])?)+(\.{</td></tr>
<tr><td>URL</td><td>/^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \</td></tr>
<tr><td>IP 地址</td><td>/((2[0-4]\d|25[0-5]|[01]?\d\d?)\.){3}(2[0-4]\d|25
/^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){
[0-9]?)$/</td></tr>
<tr><td>HTML 标签</td><td>/^<([a-z]+)([^<]+)*(?:>(.*)<\/\1>|\s+\/>)$/</td></tr>
</table>

3.4 正则表达式的代数定律

- R和S为正则表达式

- $L(RS)=L(R)L(S)$
- $L(R+S)=L(R)\cup L(S)$
- $L(R^*)=(L(R))^*$

- 若$w\in L(RS)$，则$w=xy$，且$x\in L(R)$，$y\in L(S)$
- 若$w\in L(R+S)$，则$w\in L(R)$，或者$w \in L(S)$
- 若$w\in L(R^*)$，则$w=x^*$，且$x\in L(R)$

正则表达式的代数定律

- 交换律： $R+S=S+R$
- 结合律： $(R+S)+E=R+(S+E)$、$(RS)E=R(SE)$
- 恒等律： $R+\emptyset=R$、$\varepsilon R=R\varepsilon=R$
- 零元律： $\emptyset R=R\emptyset=\emptyset$
- 分配律：$R(S+E)=RS+RE$ 、$(S+E)R=SR+ER$
- 幂等律： $R+R=R$
- 涉及Kleene 闭包的：

  $(R^*)^*=R^*$

  $\emptyset^*=\varepsilon$

  $\varepsilon^*=\varepsilon$

P63 习题6 (2)

- A是任意正则表达式，证明$(A^*)^*=A^*$

- 只须证明$L((A^*)^*)=L(A^*)$

对任意$x\in L(A^*)$， $x=A^k$，其中k为非负整数；
那么x可写成$(A^k)^1$，即与RE $(A^*)^*$匹配；
因此， $x\in L((A^*)^*)$。

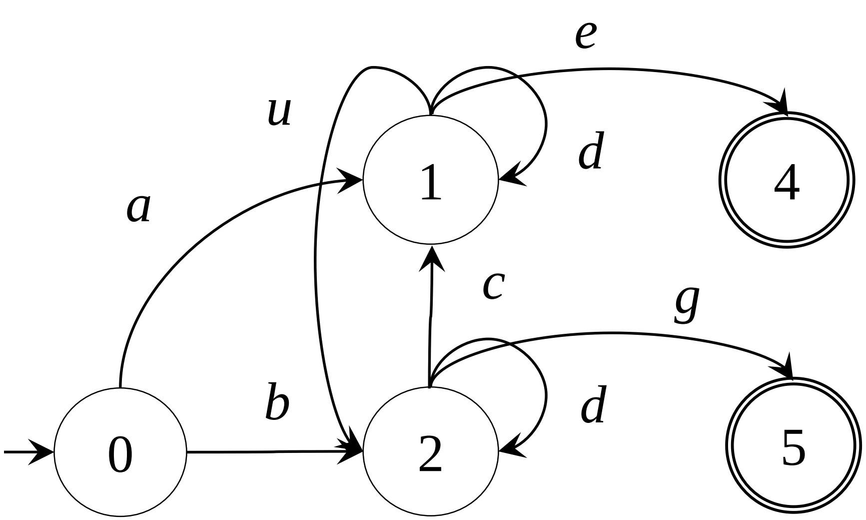
对任意$x\in L((A^*)^*)$，$x=(A^k)^j$，其中k和j为非负整数；
那么x可写成$A^{k\times j}$，其中$k\times j$为非负整数；
所以x与RE A^*匹配，从而， $x\in L(A^*)$。

小结

- 知识点：原子语言、语言运算表达式、正则表达式，正则表达式的语言、UNIX中扩展的运算符
- 知识点：正则表达式与NFA、DFA等价转换
- 知识点：正则表达式代数定律

- 习题3.1(2)；习题3.2(2)(3)；习题3.4(2)

将DFA转为RE



$\lambda(1)=\{0, 2\}; \mu(1)=\{2, 4\}$

$R_{02}+R_{01}R_{11}{}^{*}R_{12}=b+ad^{*}u$

$R_{04}+R_{01}R_{11}{}^{*}R_{14}=\emptyset+ad^{*}e$

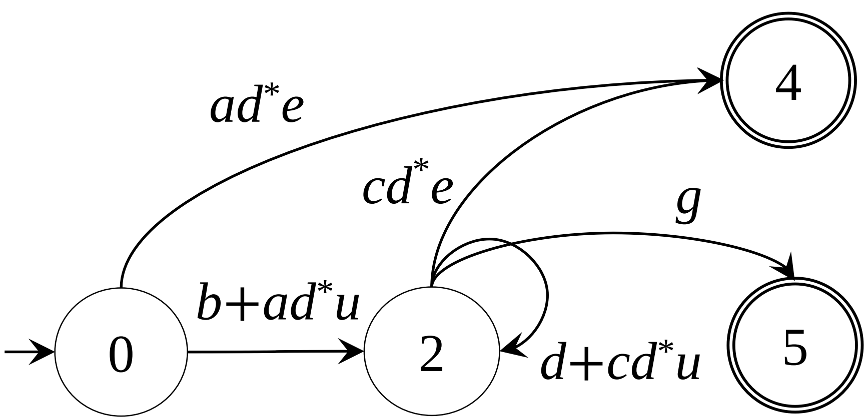
$R_{22}+R_{21}R_{11}{}^{*}R_{12}=d+cd^{*}u$

$R_{24}+R_{21}R_{11}{}^{*}R_{14}=\emptyset+cd^{*}e$

$\lambda(2)=\{0\}; \mu(2)=\{4, 5\}$

$R_{04}+R_{02}R_{22}{}^{*}R_{24}=ad^{*}e+(b+ad^{*}u)(d+cd^{*}u)^{*}cd^{*}e$

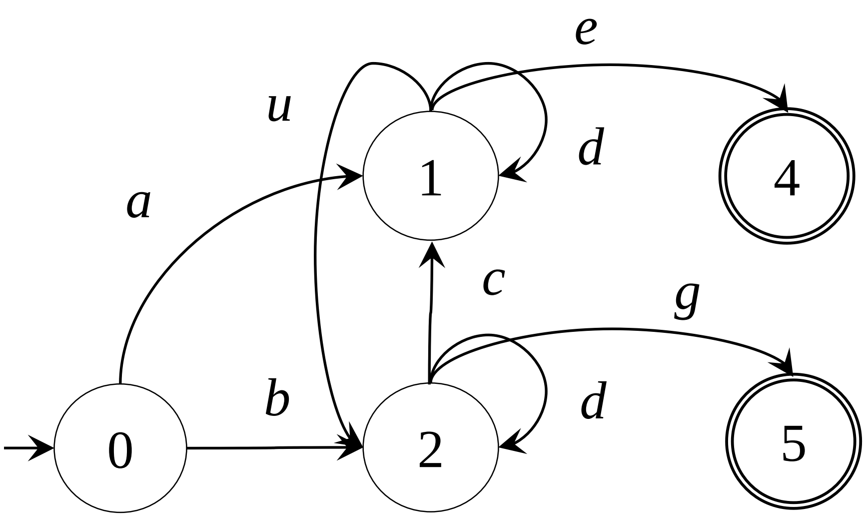
$R_{05}+R_{02}R_{22}{}^{*}R_{25}=\emptyset+(b+ad^{*}u)(d+cd^{*}u)^{*}g$



$ad^{*}e+(b+ad^{*}u)(d+cd^{*}u)^{*}cd^{*}e$
$+(b+ad^{*}u)(d+cd^{*}u)^{*}g$

$a(d+u(d+ud^{*}c)^{*}e)^{*}+b(d+cd^{*}u)^{*}g$

将DFA转为RE



$\lambda(2)=\{0, 1\};\ \mu(2)=\{1, 5\}$

$R_{01}+R_{02}R_{22}{}^{*}R_{21}=a+bd^{*}c$

$R_{05}+R_{02}R_{22}{}^{*}R_{25}=\emptyset+bd^{*}g$

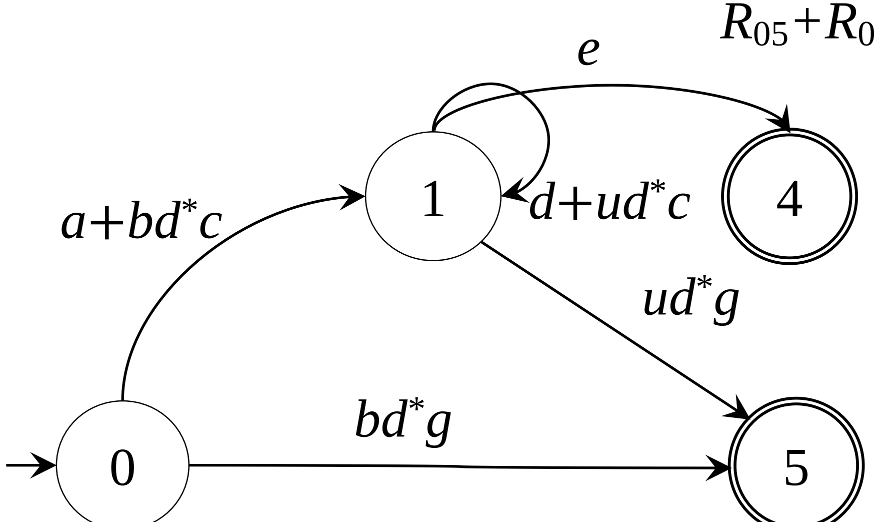
$R_{11}+R_{12}R_{22}{}^{*}R_{21}=d+ud^{*}c$

$R_{15}+R_{12}R_{22}{}^{*}R_{25}=\emptyset+ud^{*}g$

$\lambda(1)=\{0\};\ \mu(1)=\{4, 5\}$

$R_{04}+R_{01}R_{11}{}^{*}R_{14}=\emptyset+(a+bd^{*}c)(d+ud^{*}c)^{*}e$

$R_{05}+R_{01}R_{11}{}^{*}R_{15}=bd^{*}g+(a+bd^{*}c)(d+ud^{*}c)^{*}ud^{*}g$



$(a+bd^{*}c)(d+ud^{*}c)^{*}e+bd^{*}g+(a+bd^{*}c)(d+ud^{*}c)^{*}ud^{*}g$

$ad^{*}e+(b+ad^{*}u)(d+cd^{*}u)^{*}cd^{*}e+(b+ad^{*}u)(d+cd^{*}u)^{*}g$

对比分析

① $(a+bd^*c)(d+ud^*c)^*e+bd^*g+(a+bd^*c)(d+ud^*c)^*ud^*g$

② $ad^*e+(b+ad^*u)(d+cd^*u)^*cd^*e+(b+ad^*u)(d+cd^*u)^*g$

① $ad^*e+b(d+cd^*u)^*cd^*e+ ad^*u(d+cd^*u)^*cd^*e+b(d+cd^*u)^*g+ad^*u(d+cd^*u)^*g$

② $a(d+ud^*c)^*e+bd^*c(d+ud^*c)^*e+bd^*g+ a(d+ud^*c)^*ud^*g+bd^*c(d+ud^*c)^*ud^*g$

① $b(d^*cd^*ud^*)^*cd^*e$

② $bd^*c(d^*ud^*cd^*)^*e$