第4章 正则表达式

1. 正则运算与正则表达式
2. 正则表达式与FA的等价
3. 正则表达式代数定律
4. 正则表达式的应用

4.1正则表达式与正则运算

	arithmetic	theory of computation
objects	numbers	languages
tools	+, ×	∪ , ∘ , *

Regular operations be used to:

- **Design automata to recognize particular languages. (grep in Unix, Perl, text editors)**
- **Prove that certain other languages are nonregular.**

Regular operations

Definition 4.1

Let A and B be languages. We define the regular operations ***union***, ***concatenation***, and ***star*** as follows:

Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.

Concatenation: $A \cdot B = \{xy \mid x \in A \text{ and } y \in B\}$.

Star: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$.

Regular Operations

Theorem 4.1 The class of RL is closed under union operation.

Let $N_1=(Q_1, \sum,\delta_1,q_1,F_1)$ recognize A_1,$N_2=(Q_2, \sum,\delta_2,q_2,F_2)$) recognize A_2, N_1,N_2 is NFA.

Construct $N=(Q, \sum,\delta,q_0,F)$ to recognize A1UA2.

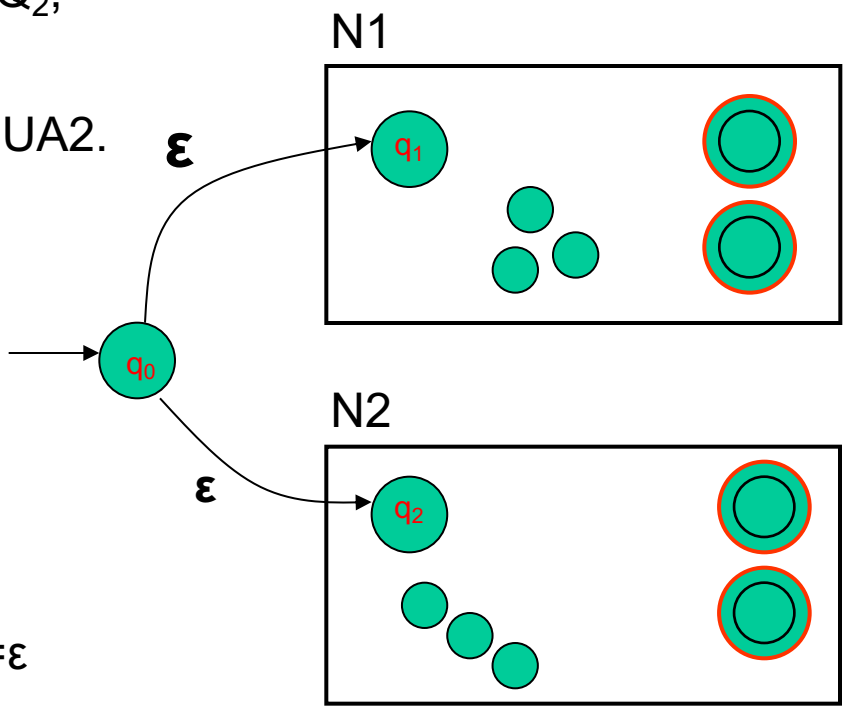
1.$Q=\{q_0\}UQ_1UQ_2$

2.The State q_0 is the start state of N.

3.The accept state $F = F_1 \cup F_2$

4.

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q\in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1,q_2\} & q=q_0 \text{ and } a=\varepsilon \\ \varnothing & q=q_0 \text{ and } a \neq \varepsilon \end{cases}$$

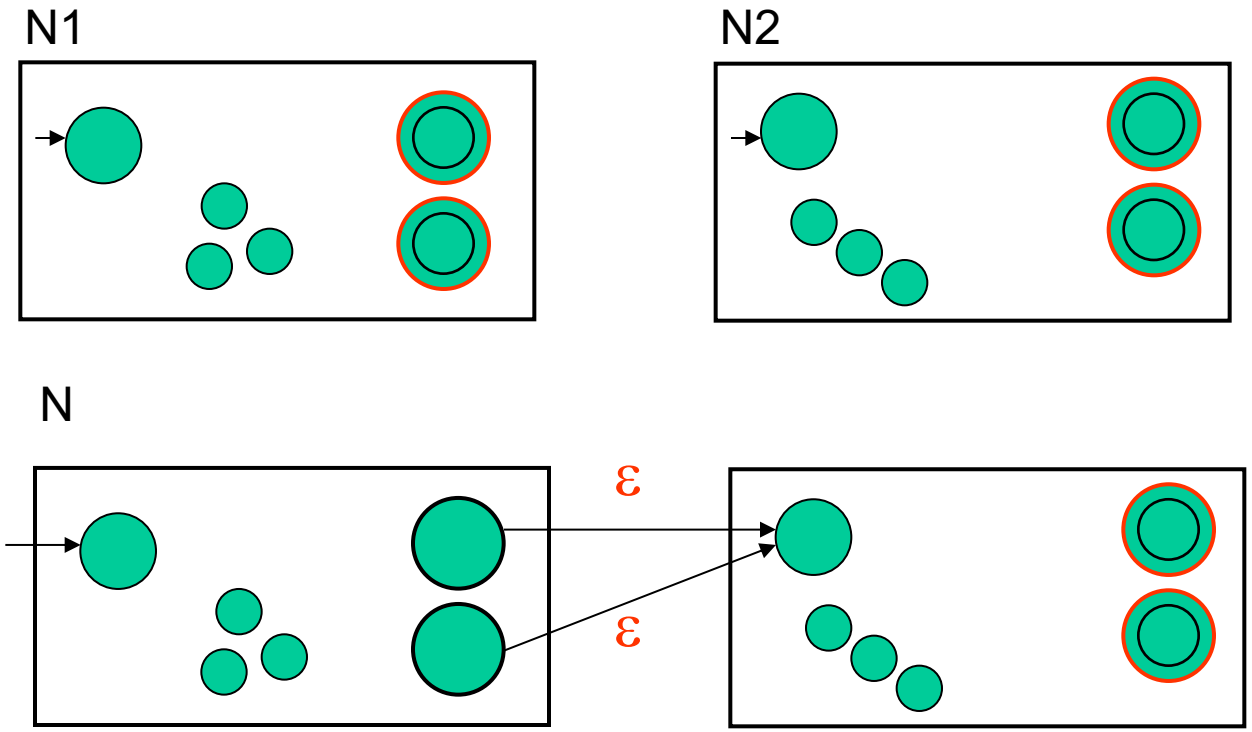


其中：$q \in Q$, $a \in \sum_\varepsilon$

Regular Operations

Theorem 4.2 The class of RL is closed under **concatenation** operation.

Proof Idea ：构造NFA N，N识别A1·A2



Regular Operations

Theorem 4.2 The class of RL is closed under concatenation operation.

PROOF

Let N1 = (Q_1, Σ, δ_1, q_1, F_1) recognize A_1, and N_2 = ($_{Q2}$, Σ, δ_2, q_2, F_2) recognize A_2.

Construct N = (Q, Σ, δ, q_1, F_2) to recognize A_1 . A_2.

1. Q = $Q_1 \cup Q_2$. The states of N are all the states of N1 and N_2.

2. The state q_1 is the same as the start state of N_1.

3. The accept states F_2 are the same as the accept states of N_2.

4. Define δ so that for any q ∈ Q and any a ∈ Σε ,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2. \end{cases}$$

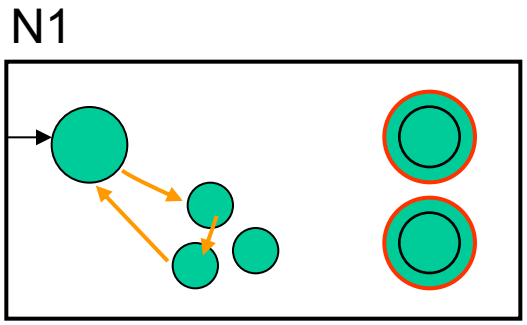
Regular Operations

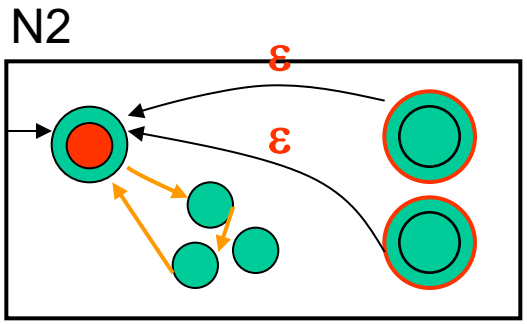
Theorem 4.3 The class of RL is closed under star operation.

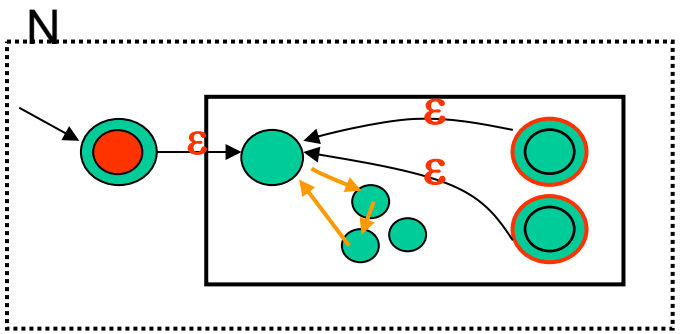
Star: $A^* = \{x_1x_2\ldots x_k \mid k>=0 \text{ and each } x_i \in A\}$

Proof Idea: Construction of N to recognize A^*

- A^* 可分解成若干个片段x_i，且$x_i \in A$，所以，接受状态应返回起始状态；
- A^* 包含 ε ，所以，起始状态起应该是接受状态；
- **问题：N2识别A*吗？**







Regular Operations

Theorem 4.3 The class of RL is closed under star operation.

PROOF

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.

Construct $N = (Q,\Sigma,\delta,q_0,F)$ to recognize A_1^*.

1. $Q = \{q_0\} \cup Q_1$.

The states of N are the states of N_1 plus a new start state.

2. The state q_0 is the new start state.

3. $F = \{q_0\} \cup F_1$.

The accept states are the old accept states plus the new start state.

4. Define δ so that for any $q \in Q$ and any $a \in \Sigma\varepsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

Regular Expressions

Definition 4.2

Given an alphabet Σ, R is a regular expression if

1. R = a, with $a \in \Sigma$; denoting the languages {a } .
2. R = ε ; denoting the languages {ε } .
3. R = ∅ ; denoting the languages ∅.
4. R = $(R_1 + R_2)$, with R_1 and R_2 regular expressions; denoting the languages $L(R_1) \cup L(R_2)$.
5. R = $(R_1 \bullet R_2)$, with R_1 and R_2 regular expressions; denoting the languages L(R1) L(R2).
6. R = (R_1^*), with R_1 a regular expression; denoting the languages $L(R_1)$ *.

Regular Expressions

Example 4.1 What is the language defined by r

$$r = (a + b)^*(a + bb)\ .$$

a → { a }, b → { b }

a+b → { a } ∪ { b } = {a, b}

bb → { b } { b } = { bb }

a + bb → {a } ∪ {bb } = { a, bb }

(a +b)* → { a ,b }*

(a +b)* (a +bb) → { a ,b }* { a, bb }

L(r) = { a, bb, aa, abb, ba, bbb, ... }

Regular Expressions

Example 4.2 What is the language defined by r

$$r = (aa)^*(bb)^* b$$

$$L(r) = (\{a\}\{a\})^* (\{b\}\{b\})^* \{b\}$$

$$=(\{aa\})^*(\{bb\})^*\{b\}$$

$$= \{aa\}^* \{bb\}^*\{b\}$$

$$= \{a^{2n}b^{2m+1} \mid n \geq 0, m \geq 0\}$$

Regular Expressions

Example 4.3 Write a regular expression for the set of strings that consist of alternating 0's and 1's.

Partition :

$010101...0101 \longrightarrow (01)^*$

$101010...1010 \longrightarrow (10)^*$

$0101010...1010 \longrightarrow 0(10)^* \text{ or } (01)^*0$

$101010...10101 \longrightarrow 1\,(01)^* \text{ or } (10)^*1$

The regular expression :

$$(01)^*+(10)^*+0(10)^*+1(01)^* = (\varepsilon +1)(01)^*(\varepsilon +0)$$

Regular Expressions

Example 4.4 Design regular expression for L ,
L={w | w∈ {0, 1}* and w has no pair of consecutive 0's } .

Partition :

no 0 ⟶ 1^*

one 0 ⟶ 1^*01^*

more 0's ⟶ $(1^* 011^*)^* (0+\varepsilon)$

$r1 = (1*011*)^* (0+\varepsilon)+1*(01^* +\varepsilon)$

或 $r2 = (1+01)^*(0+\varepsilon)$

思考题：

1. r1 = r2?
2. L(r) = {w ∈ Σ*| w has at least one pair of consecutive zeros}. R =?

构造正则表达式的一般性规则

1. **语言只含一个字符串**：字符串作为正则表达式。

   如：正则表达式00和11表示语言{00}和{11}。

2. **语言含多个字符串的连接得到的串**：字符串的连接作为正则表达式。

   如：正则表达式00和11表示语言{00}和{11}，那么，正则表达式0011表示语言{0011}。

3. **语言含零次或多次出现的串**：串的闭包作为正则表达式。如：含01的零次或多次出现的串，正则表达式为(01)*。

构造正则表达式的一般性规则

4. **语言中的串有多种可能的形式**：用+运算符表达多种可能性（相当于语言的并）。

   如：(01)*+(10)*+0(10)*+1(01)*。

5. **语言的串中含可有、可无的子串**：用ε和该子串的并与串的其余部分连接。

   如：(ε +1)(01)*(ε +0)

Regular Expressions

Example 4.5 In the following instances, we assume that the alphabet Σ is $\{0,1\}$.

1. $0^*10^* = \{w|$ w contains a single 1$\}$.
2. $\Sigma^*1\Sigma^* = \{w|$ w has at least one 1$\}$.
3. $\Sigma^*001\Sigma^* = \{w|$ w contains the string 001 as a substring$\}$.
4. $1^*(01^+)^* = \{w|$ every 0 in w is followed by at least one 1$\}$.
5. $(\Sigma\Sigma)^* = \{w|$ w is a string of even length$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w|$ the length of w is a multiple of 3$\}$.
7. $01 + 10 = \{01, 10\}$.

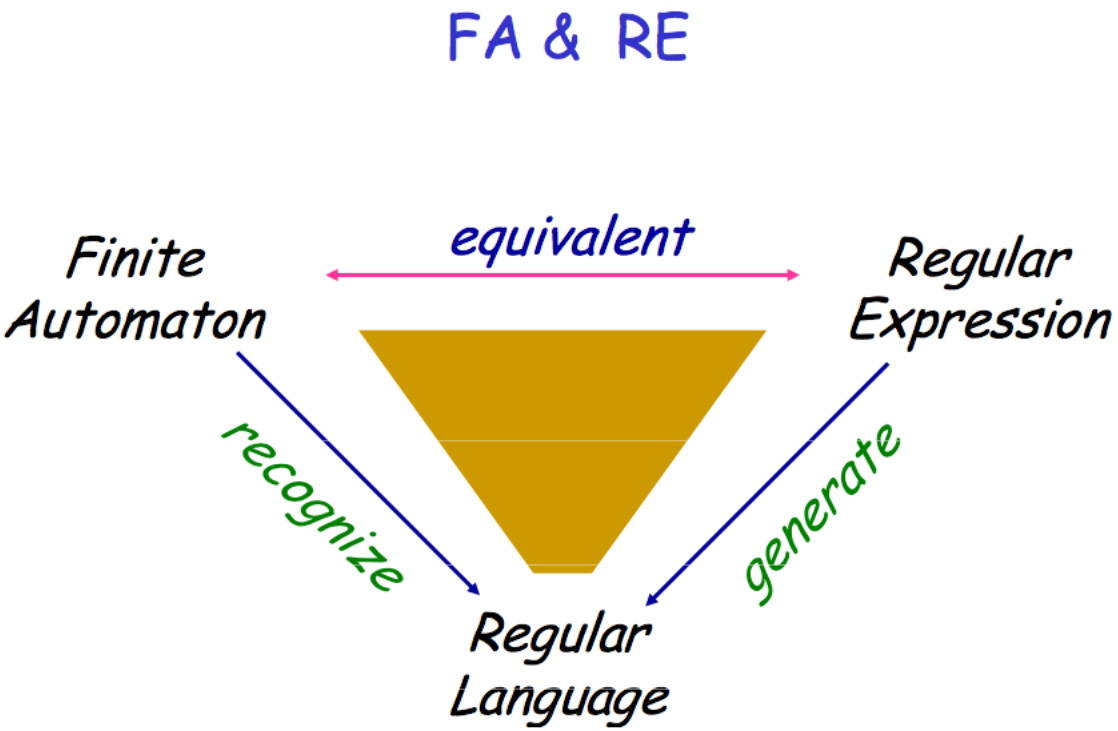
Regular Expressions

Example 4.5 （续）

8. $0\Sigma^*0 + 1\Sigma^*1 + 0 + 1 = \{w|$ w starts and ends with the same symbol$\}$.
9. $(0 + \varepsilon)1^* = 01^* \cup 1^*$.
10. $(0 + \varepsilon)(1 + \varepsilon) = \{\varepsilon, 0, 1, 01\}$.
11. $1^*\emptyset = \emptyset$.
12. $\emptyset^* = \{\varepsilon\}$.
13. $R + \emptyset = R$.
14. $R \bullet \varepsilon = R$.
15. $R \bullet \emptyset = \emptyset$.

思考题：$\emptyset^* = ?$，$\emptyset^0 = ?$，$\emptyset^i = ?$，其中$i \geq 1$

4.2 正则表达式与FA的等价关系



Equivalence with FA

Theorem 4. 4 **A language is regular if and only if some regular expression describes it.**

Lemma 4.5 **If a language is described by a regular expression, then it is regular.**

（正则表达式表示的语言是正则语言，RE⇒FA）

Lemma 4.6 **If a language is regular, then it is described by a regular expression.**

（正则语言可以用正则表达式表示，FA⇒RE）

Equivalence with FA

Lemma 4.5 **If a language is described by a regular expression, then it is regular.**

（正则表达式表示的语言是正则语言，RE⇒FA）

Proof Idea:

RE R describes L(R) , L(R) is RL. （求证）

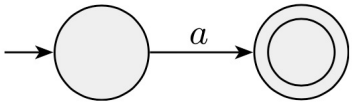
NFA N recognizes L(N) , L(N) is RL. （已知）

How to convert RE R to NFA N ?

Equivalence with FA

Proof1: Let's convert R to NFA N. We consider the six cases in the formal definition of regular expressions.

1. R = a for some a ∈ Σ . Then L(R) = {a}, and the following NFA recognizes L(R).

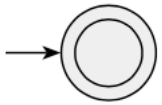


- Note that this machine fits the definition of an NFA but not that of a DFA。
- Formally, N = { $\{q_1, q_2\}$, Σ, δ, q_1, $\{q_2\}$ } , where we describe δ By saying that $\delta(q_1,a)=\{q_2\}$ and that $\delta(r , b)=\phi$ for $r \neq q_1$ or $b \neq a$.

Equivalence with FA

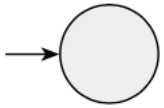
Proof1（续）:

2. R = ε. Then L(R) = {ε}, and the following NFA recognizes L(R).



Formally, N = { $\{q_1\}$, Σ, δ, q_1, $\{q_1\}$ } , where δ(r, b) =ϕ for any r and b.

3. R = ∅. Then L(R) =ϕ, and the following NFA recognizes L(R).



Formally, N ={ {q},Σ,δ,q,ϕ}, where δ(r,b) =ϕ for any r and b.

Equivalence with FA

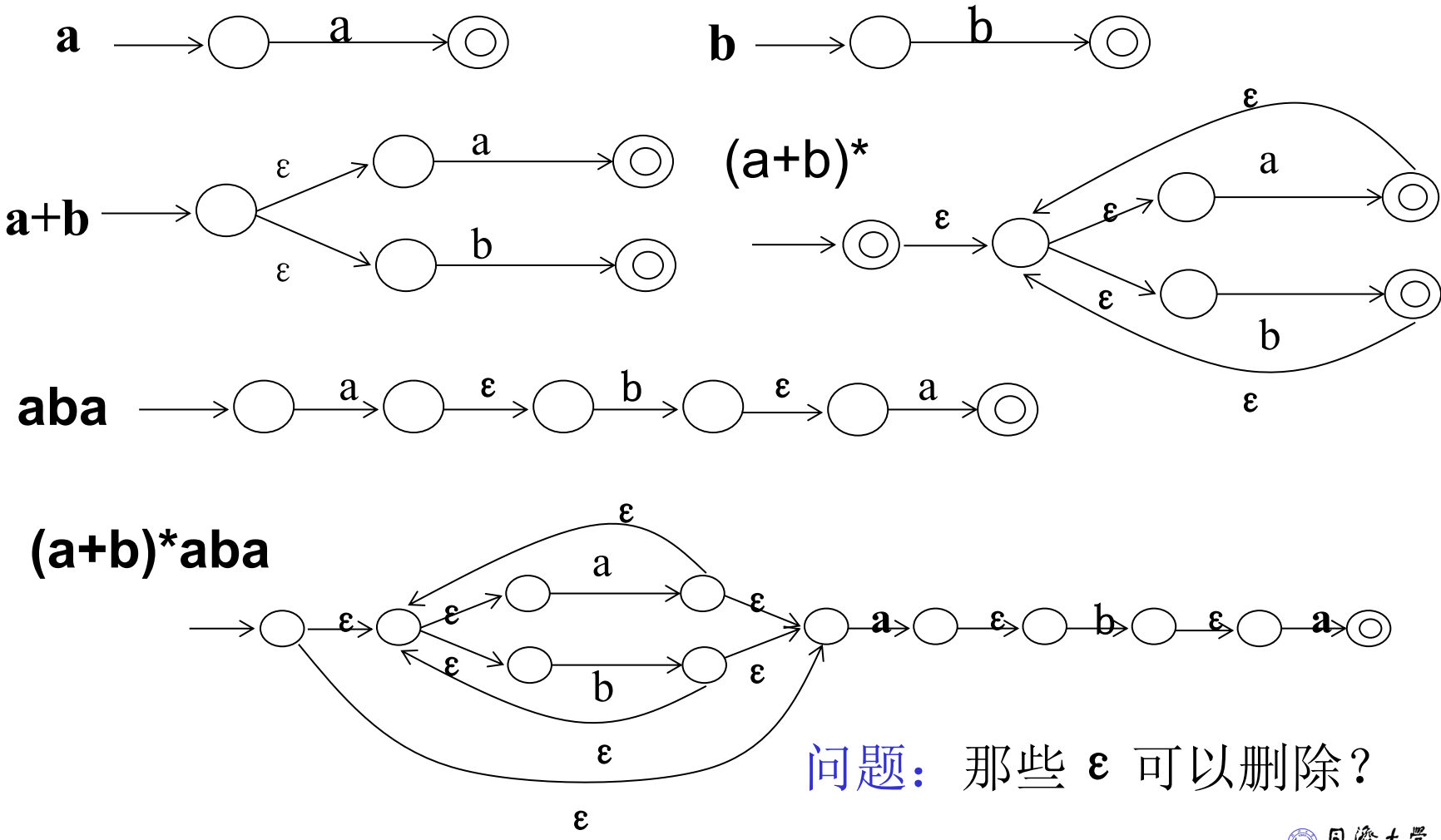
Proof1（续）: Let's convert R into an NFA N. We consider the six cases in the formal definition of regular expressions.

4. $\mathbf{R = R_1 + R_2}$
5. $\mathbf{R = R_1 . R_2}$
6. $\mathbf{R = R_1^*}$

We construct the NFA for R from the NFAs for R_1 and R_2 (or just R_1 in case 6) and the appropriate closure construction. THeorem4.1, 4.2, 4.3.

Equivalence with FA

Example 4.6 将正则表达式（a+b）*aba转化为NFA。

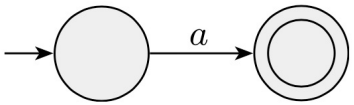


问题：那些 ε 可以删除？

Equivalence with FA

Proof2：对正则表达式的运算符个数n进行归纳证明。

1. n=0，R是单个字母，如R=a,造自动机识别它



2. 设定理对n=k成立

3. n=k+1时，r有3种情况$r_1+r_2, r_1.r_2, r_1^*$。对最后计算的一个符号分情况（连接，并，星）用定理（Theorem4.1,4.2 , 4.3）即得。

Equivalence with FA

Lemma 4.6 If a language is regular, then it is described by a regular expression.

（正则语言可以用正则表达式表示，FA⇒RE）

Proof Idea:

方法一：GNFA法

① RL有DFA M识别，把DFA转化成广义的GNFA。

② 把广义的GNFA转化成正则表达式 RE 。

方法二：R_{ij}^{k} 迭代法

GNFA

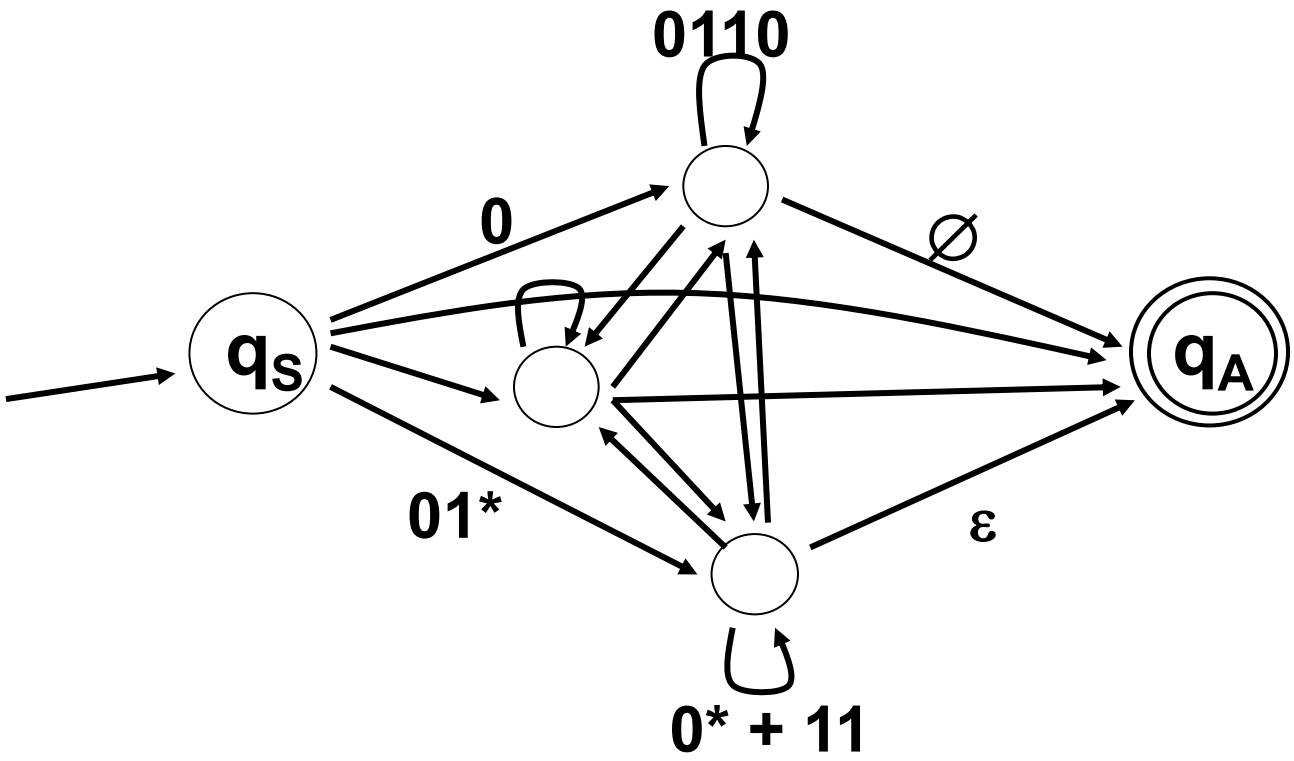
Definition 4.3

A Generalized nondeterministic finite automaton(GNFA) is a 5-tuple$(Q, \Sigma, \delta, q_{start}, q_{accept})$,where

1. Q is the finite states.
2. Σ is the input alphabet.
3. q_{start} is the start state.
4. q_{accept} is the accept state.
5. $\delta:(Q-\{q_{accept}\})\times(Q-\{q_{start}\}) \rightarrow \mathcal{R}$ is the transition function.

- $\mathcal{R}$ is the set of regular expressions over Σ.
- $\delta(q_i, q_j)=R \Leftrightarrow$ from q_i to q_j has the RE R as the label, $R \in \mathcal{R}$.

GNFA



A generalized nondeterministic finite automaton

GNFA的特点

1. The interior Q-{q_{accept},q_{start}} is fully connected by δ
2. From q_{start} only ‘outgoing transitions’
3. To q_{accept} only ‘ingoing transitions’
4. Impossible $q_i \rightarrow q_j$ transitions are “$\delta(q_i,q_j) = \varnothing$”
5. $q_{accept} \neq q_{start}$

Observation: This GNFA recognizes the language L(R)

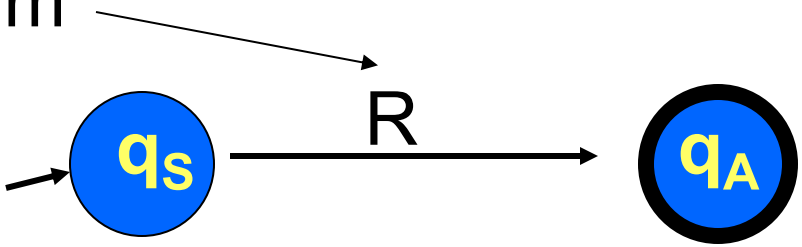
- **GNFA与一般NFA的唯一区别**: GNFA的边推广为 RE（子自动机），而不是单个的字母或ε。
- **为了简洁易读，通常省略不可能的转移**$(q_i,q_j) = \varnothing$。

Equivalence with FA

Proof Idea (given a DFA M):

减少状态（减少跳转标号），　扩大广义边（子程序）

1. Construct an equivalent GNFA M’ with k≥2 states
2. Reduce one-by-one the internal states until k=2
   逐个等价地减少状态，把内部的正则表达式变大
3. This GNFA will be of the form



This regular expression R will be such that L(R) = L(M)

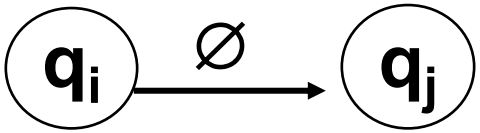
Equivalence with FA

证明（方法一：GNFA法）

Let M have k states $Q=\{q_1,\ldots,q_k\}$

1. Add two states q_{accept} and q_{start} (加首尾)
   - Connect q_{start} to earlier q_1:
   - Connect old accepting states to q_{accept}

2. Complete missing transitions by ∅ （补空边）



3. Join multiple transitions (并标号)

Equivalence with FA

4. Rip the internal states, one by one (删中间)

① Removing state $q_{rip} \in Q - \{q_{start}\} - \{q_{accept}\}$:

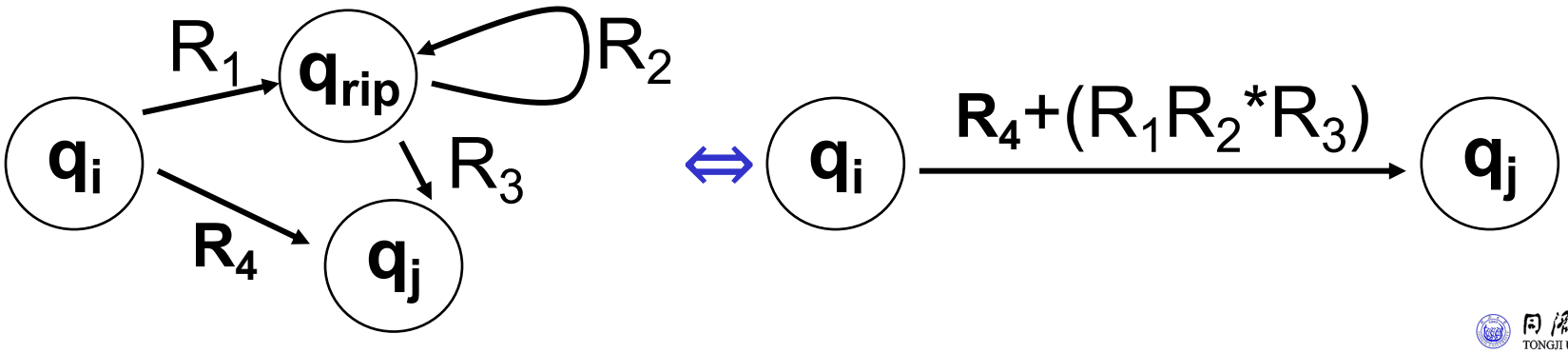
$Q' = Q - \{q_{rip}\}$ 逐步减少内态

① Changing the transition function δ by

$$\delta'(q_i,q_j) = \delta(q_i,q_j) + \delta(q_i,q_{rip})(\delta(q_{rip},q_{rip}))^*\delta(q_{rip},q_j)$$

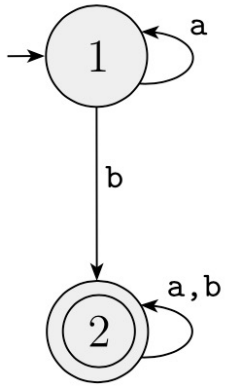
for every $q_i \in Q' - \{q_{accept}\}$ and $q_j \in Q' - \{q_{start}\}$

5. If k > 2 goto 4, else return R

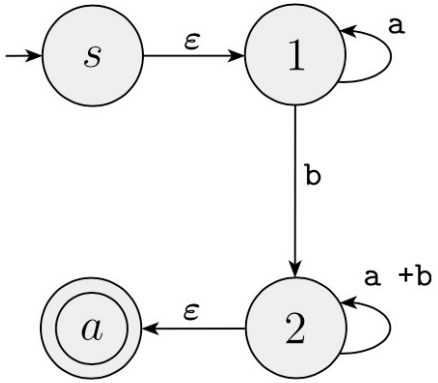


Equivalence with FA

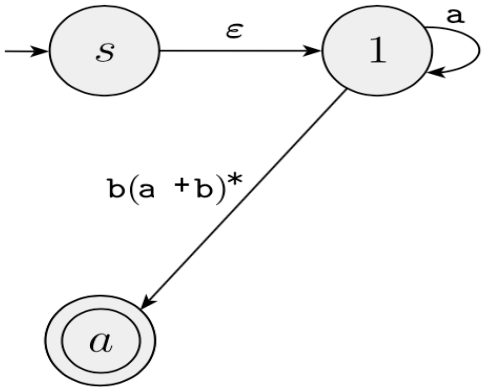
Example 4.7 Converting a two-state DFA to an equivalent regular expression.



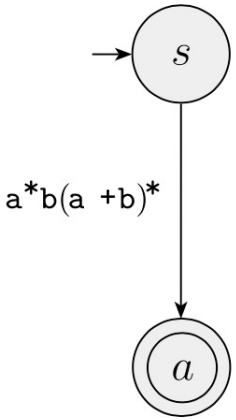
(a)



(b)



(c)



(d)

Equivalence with FA

证明（方法二）

设 DFA $M(\{q_1,q_2,\ldots,q_n\}, \sum, \delta, q_1, F)$ 接受语言 L，对M的状态进行了编号。在此基础上，引入记号R_{ij}^k，它是字符串的集合，定义如下：

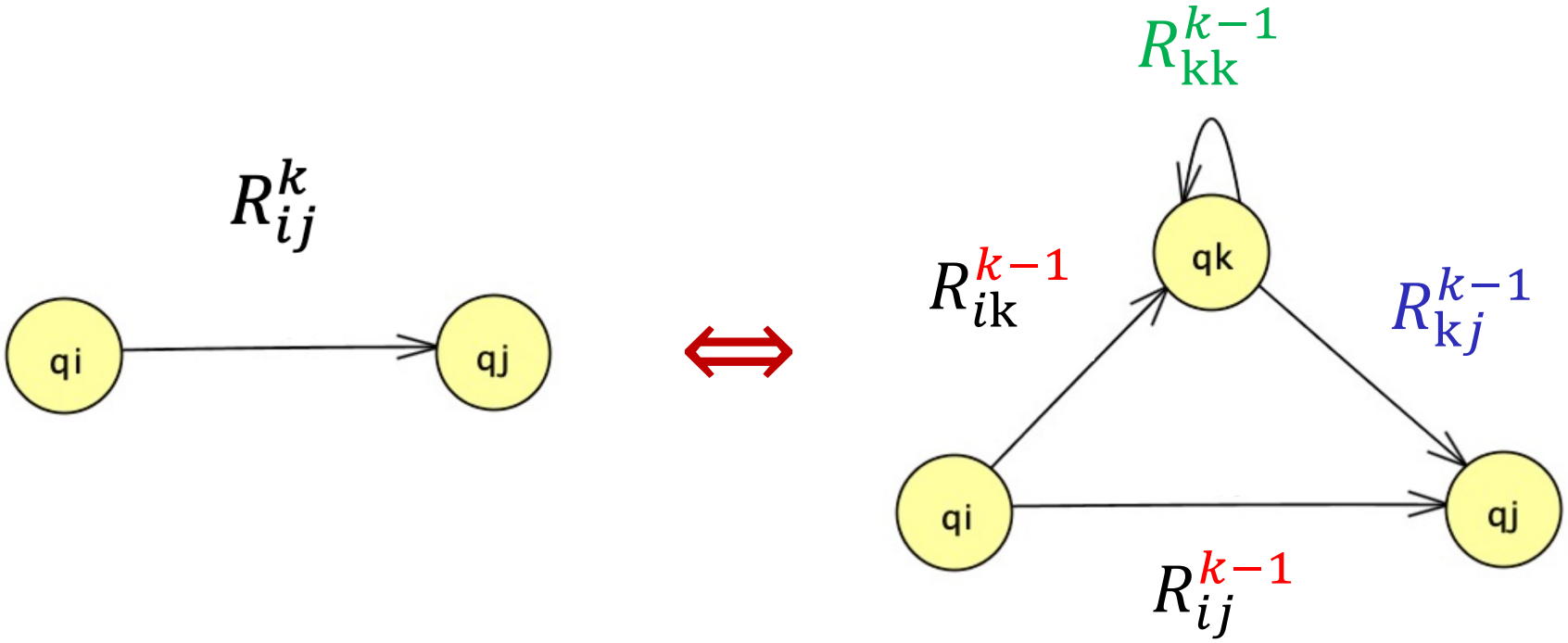
$R_{ij}^k = \{ x \mid \delta(q_i, x) = q_j$，且中间不经过编号大于k的状态，但i、j可以大于k，$x \in \Sigma^* \}$

注意：x是字符串

k的意义是：对于x的一切非空真前缀y，$\delta(q_i,y) = q_m$，其中$m \le k$。

Equivalence with FA

证明（方法二） 基本思想：引入R_{ij}的递归定义。



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

Equivalence with FA

R_{ij}^k的递归定义如下：

① $R_{ij}^0 = \{a \mid a \in \sum, \text{且 } \delta(q_i,a) = q_j\}$ （i≠j）

② $R_{ij}^0 = \{a \mid a \in \sum, \text{且 } \delta(q_i,a) = q_j\} \cup \{\varepsilon\}$ （i = j）

③ $R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1}$ （k=1,2,…,n）

解释：

① k=0 & i ≠ j时，q_i到q_j只能一步到达，∴a是单个字符；

② k=0 & i = j时，除了a外，还包括ε， ∵对于任意状态q_i，都有$\delta(q_i, \varepsilon) = q_j$。

③ R_{ij}^k分两种情况讨论：见下一页。

Equivalence with FA

③ R_{ij}^{k}分两种情况讨论（续）

Case 1：若M从q_i出发，读字符串x，到达q_j的过程中，不经过编号大于k-1的任何状态，则x应在R_{ij}^{k-1}中，当然也在R_{ij}^{k}中，所以它应是R_{ij}^{k}的一部分；

Case 2：若经过编号等于k的状态一次或多次，状态变化序列如下，其中 “…” 处出现的状态编号均小于k。

$$q_i \cdots q_k \cdots q_k \cdots q_k \cdots q_j$$

从q_i …读过x的子串属于R_{ik}^{k-1}，从$q_k \ldots q_k \ldots$读过的x的子串属于$(R_{kk}^{k-1})^*$，从最后一个q_k开始，$q_k \ldots$读过的x的子串属于R_{kj}^{k-1}。因此，x应在$R_{ik}^{k-1}(R_{kk}^{k-1})^*R_{kj}^{k-1}$中。

综合case1、2有：$R_{ij}^{k} = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^*R_{kj}^{k-1}$

Equivalence with FA

回到原点：**Lemma 4.6 If a language is regular, then it is described by a regular expression.**

即，已知DFA M识别语言L(M)，有：

$$\mathrm{L}(M) = \bigcup_{q_f \in F} R_{1f}^{n}$$

L(M)就是DFA M识别的语言，其中q_f是可接受状态。

问题：如何将L(M)用正则表达式表示呢？

如果对于任意的R_{1f}^{n}，存在正则表达式r_{1f}^{n}，则即可。

Equivalence with FA

现证，对于任何R_{ij}^{k}，存在正则表达式r_{ij}^{k} 代表R_{ij}^{k} 。

归纳基础： k=0。 R_{ij}^{0}是一个有穷集合，其中每个元素是∑中的符号或ε，因此r_{ij}^{0}可以写成$a_1+a_2+\ldots+a_p$ (i≠j)或$a_1+a_2+\ldots+a_p+\varepsilon$ (i=j)的形式。这里，$\{a_1,a_2,\ldots,a_p\}$是使$\delta(q_i,a)=q_j$的一切符号a的集合。

归纳步骤：设对m<k的一切m，都已求出正则表达式r_{ij}^{m}代表R_{ij}^{m}，现在考虑m=k。根据R_{ij}^{k}递归定义，存在正则表达式

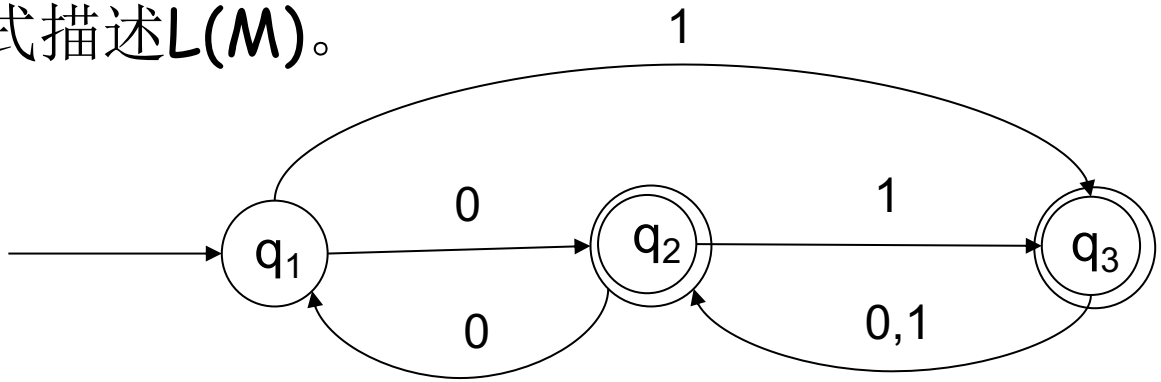
$$r_{ij}^{k}= r_{ik}^{k-1}(r_{kk}^{k-1})^{*}r_{kj}^{k-1}+r_{ij}^{k-1}$$

代表R_{ij}^{k}。

证毕。

Equivalence with FA

Example 4.8 给定一个DFA M，按照证明方法二 构造一个正则表达式描述L(M)。



分析：

1. 共3个状态，n=3；k＝0，1，2，3；
2. 根据递归公式求r_{ij}^k，其中k=0，1，2.（见下页）

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

3. 由状态图可知$L(M) = R_{12}^3 \cup R_{13}^3$，问题是求$r_{12}^3 + r_{13}^3$代表L(M)。

Equivalence with FA

	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	ε+00	(00)*
r_{23}^k	1	1+01	0*1
r_{31}^k	∅	∅	(0+1)(00)*0
r_{32}^k	0+1	0+1	(0+1)(00)*
r_{33}^k	ε	ε	ε+(0+1)0*1

$r_{12}^3 = r_{12}^2 + r_{13}^2 \left(r_{33}^2 \right)^* r_{32}^2$

$r_{13}^3 = r_{13}^2 + r_{13}^2 \left(r_{33}^2 \right)^* r_{33}^2$

$r = r_{12}^3 + r_{13}^3$

$= \cdots\cdots$

4.3 正则表达式代数定律

主要目的：正则表达式的变换。

基本问题：在定义相同语言的意义下，两个表达式等价。

基本思路：类比算术代数，比如：加法交换律、乘法分配律等

用途：正则表达式化简、变换、推算

4.3 正则表达式代数定律

1. 交换律与结合律

设L、M和N是正则表达式，则：

① L + M = M + L，并的交换率

② (L + M) + N = L + (M + N)，并的结合律

③ (LM)N = L(MN)， 连接的结合律

3. 分配律

L(M + N) = LM + LN，连接对于并的左分配律
(M + N)L = ML + NL，连接对于并的右分配律

4. 幂等律

幂等：一个运算符作用到两个相同的参数值，结果还是那个值。
L + L = L，并的幂等律

4.3 正则表达式代数定律

5. 与闭包有关的定律

① $(L^*)^* = L^*$，表达式闭包的闭包，不改变该语言

② $L(\Phi^*) = L(\varepsilon)=\{\varepsilon\}$，$\Phi$的闭包与$\varepsilon$等价，可以写$\Phi^* =\varepsilon$

③ $\varepsilon^* = \varepsilon$ ，多个空串连接，还是空串

④ $L^+ = LL^* = L^*L$，L^+表示1个或多个L

⑤ $L^* = L^+ + \varepsilon$

4.4 正则表达式的应用

- UNIX中的正则表达式
- 词法分析
- 查找文本中的模式

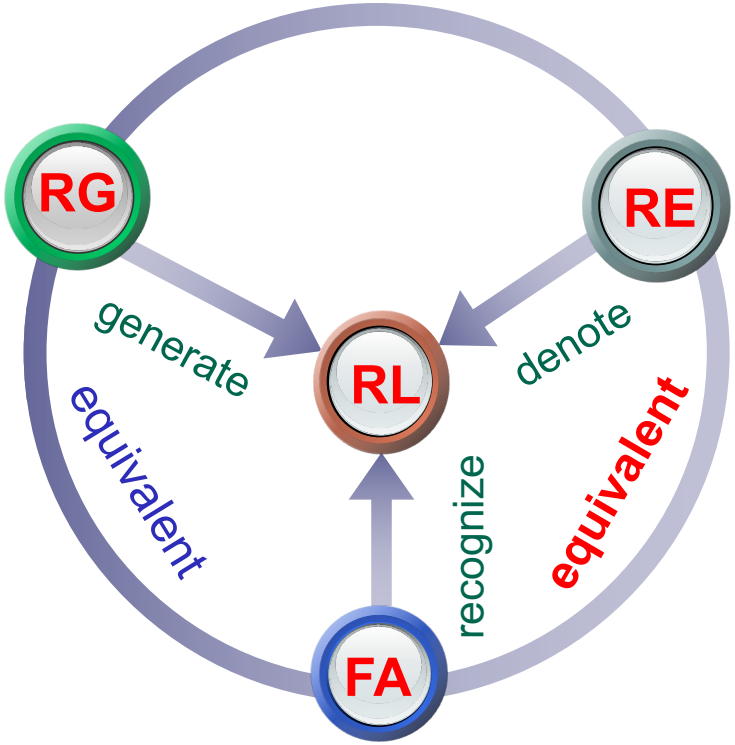
参见：

1. 霍普克罗夫特. 自动机理论、语言和计算导论：第3版[M]. 机械工业出版社，2008. p73-77
2. 陈火旺. 程序设计语言编译原理.第3版[M]. 国防工业出版社，2000. 第三章词法分析

Review

- Review:
  - Language & Regular language
    - A language is a set of strings;
    - language is called a regular language if some finite automaton recognizes it.
  - Regular Operations
    - Theorem 4.1,4.2 , 4.2.
  - Regular Expressions
    - The value of a Regular expression is a language.
    - Theorem 4.4 RL ↔ RE (describes it)
    - Lemma 4.5 If a language is described by a regular expression , then it is regular
    - Lemma 4.6 If a language is regular, then it is described by a regular expression.

Review



- RL: Regular Language
- RG: Regular Grammar
- RE: Regular Expression
- FA: Finite Automaton