编译原理

武汉大学计算机学院
编译原理课程组

前述内容回顾

文法　$G[S]=(V_N, V_T, P, S)$　　文法分类

推导　凑规则

语言　$L(G[S])=\{u \mid S \overset{*}{\Rightarrow} u\text{，且}u \in V_T^*\}$　　句型、句子

与语法分析有关的概念：

最左（右）推导、规范推导　语法树和二义性　递归

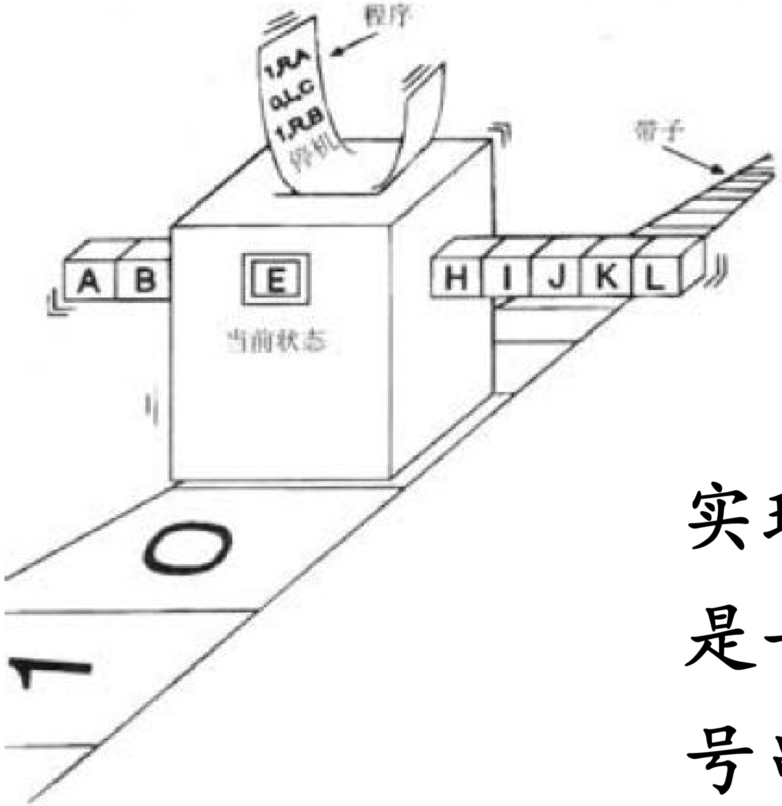
文法的实用限制

本章内容简介

- DFA、NDFA
- NDFA到DFA的转换
- 正规文法与FA
- 正规表达式与FA

语言的描述

第3章 有穷自动机



自动机是一种能进行运算并实现自我控制的装置，计算机就是一部自动机。自动机是描述符号串处理的强有力的工具。

3.1 有穷自动机的形式定义

1. 确定的有穷自动机DFA　DFA=(Q，Σ，t，q_0，F)

Q —— 有穷非空的状态集。

∑—— 有穷的输入字母表。

q_0 —— ∈Q，是开始状态。

F —— ⊆Q，非空终止状态集合。

t —— 单值映射$Q\times\Sigma\rightarrow Q$。　$t(q,x)=q'$

“确定”：当前状态和下一个输入字符惟一地确定了后继状态。

3.1 FA的形式定义——DFA

2. FA的表示：

①状态转换表

②状态转换图

DFA的表示举例——状态转换表

DFA A=({q_1，q_2，q_3，q_4}，{0，1}，t，q_1，{q_3，q_4})

映射为：

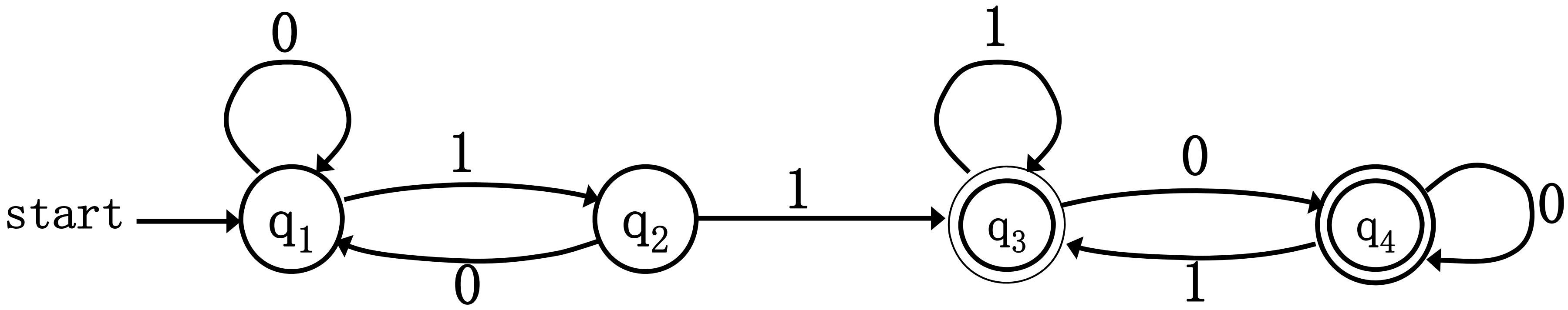
t 状态 \ 字母	0	1
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_4	q_3
q_4	q_4	q_3

DFA的表示举例——状态转换图

DFA A=({q_1, q_2, q_3, q_4},

{0, 1}, t, q_1, {q_3, q_4})

状态 \ t 字母	0	1
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_4	q_3
q_4	q_4	q_3



3.1 FA的形式定义——DFA的扩充

DFA=（Q，Σ，t，q_0，F）扩充的映射

$$t: Q \times \Sigma^* \rightarrow Q$$

定义为 (1) $t(q, \varepsilon)=q$

(2) $t(q, a\alpha)=t(t(q, a), \alpha)$

其中$q \in Q$，$a \in \Sigma$，$\alpha \in \Sigma^*$。

DFA映射的扩充，使得DFA可以描述对符号串的识别。

如果$t(q_0, \alpha) \in F$，则α可被DFA接受（或识别）。

被DFA A识别的符号串集合，记为L(A)。

有穷自动机识别的符号串举例

DFA $A=(\{q_1, q_2, q_3, q_4\},$

$\{0, 1\}, t, q_1, \{q_3, q_4\})$

t 字母 / 状态	0	1
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_4	q_3
q_4	q_4	q_3

$t(q_1, 0011)=t(t(q_1,0),011)$

$=t(q_1,011)$

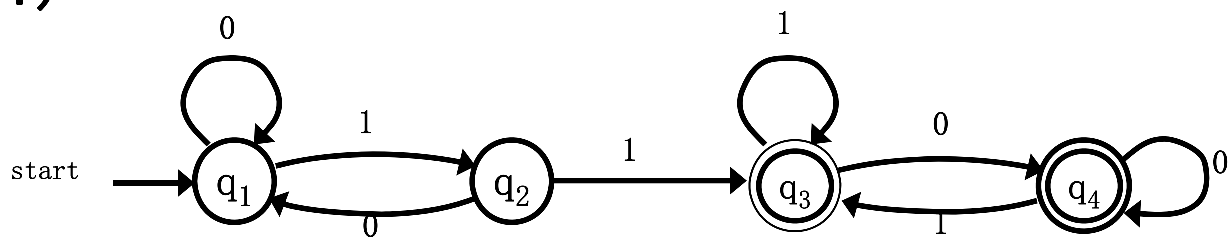
$=t(t(q_1,0),11)$

$=t(q_1,11)$

$=t(t(q_1,1),1)$

$=t(q_2,1)$

$=q_3\in F$



3.1 FA的形式定义——FA的等价性

如果两个有穷自动机A_1和A_2满足

$$L(A_1)=L(A_2)$$

则称自动机A_1和A_2是等价的。

FA的等价性举例

DFA A=($\{q_0, q_1\}$, $\{a, b\}$, t, q_0, $\{q_0\}$)

$t(q_0, a)=q_1$, $t(q_1, b)=q_0$

DFA B=($\{q_0', q_1', q_2'\}$, $\{a, b\}$, t', q_0', $\{q_0', q_2'\}$)

$t'(q_0', a)=q_1'$, $t'(q_1', b)=q_2'$, $t'(q_2', a)=q_1'$

$$L(A)=L(B)=\{(ab)^n \mid n\geq 0\}$$

3.1 有穷自动机的形式定义

非确定的有穷自动机NDFA　　NDFA=(Q，∑，t，Q_0，F)

Q —— 有穷非空的状态集。

∑ —— 有穷的输入字母表。

Q_0 —— ⊆Q，是开始状态集。

F—— ⊆Q，非空终止状态集合。

t —— 多值映射 $Q\times\Sigma\rightarrow 2^Q$。

$$t(q,x)=\{q_1,q_2,\ldots,q_n\}$$

NDFA举例

NDFA=({0,1,2,3}，{x,y}，t，{0}，{1})

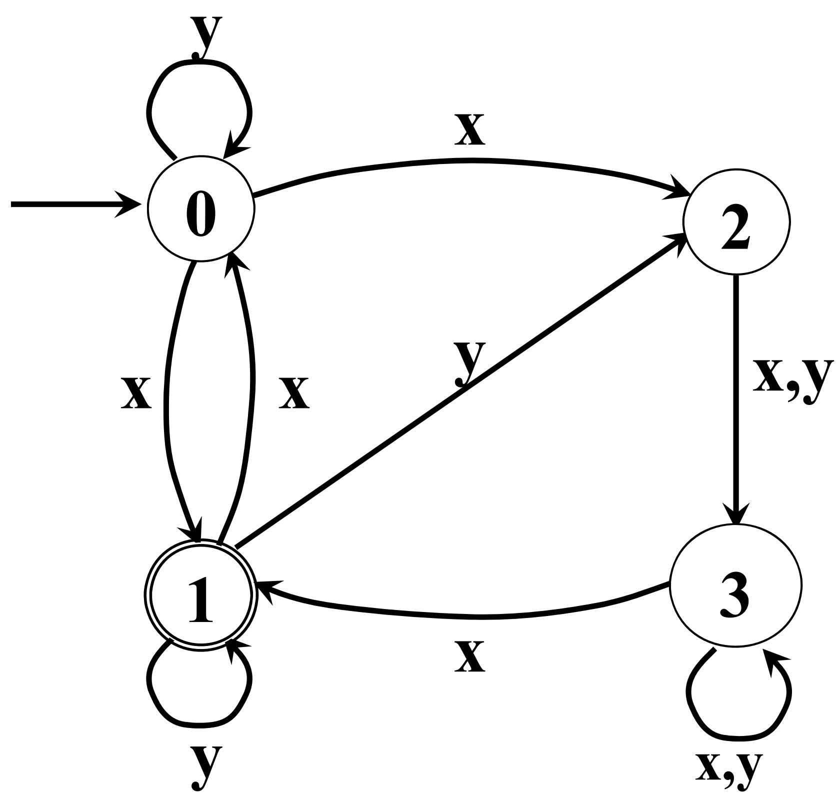
其中t为

t(0，x)={1,2}　　t(0，y)={0}

t(1，x)={0}　　t(1，y)={1,2}

t(2，x)={3}　　t(2，y)={3}

t(3，x)={1,3}　　t(3，y))={3}



3.1 FA的形式定义——NDFA的扩充

NDFA=（Q，Σ，t，Q_0，F）扩充的映射

$t: Q \times \Sigma^* \rightarrow 2^Q$

定义为 (1) $t(q,\varepsilon)=q$

(2) $t(q,a\alpha)=t(q_1,\alpha)\cup t(q_2,\alpha)\cup \ldots \cup t(q_n,\alpha)$

其中$a\in\Sigma$, $\alpha\in\Sigma^*$，$t(q,a)=\{q_1, q_2, \ldots, q_n\}$。

如果 $q\in t(q_0, \alpha)$，$q_0\in Q_0$，$q\in F$,

则α可被NDFA接受。

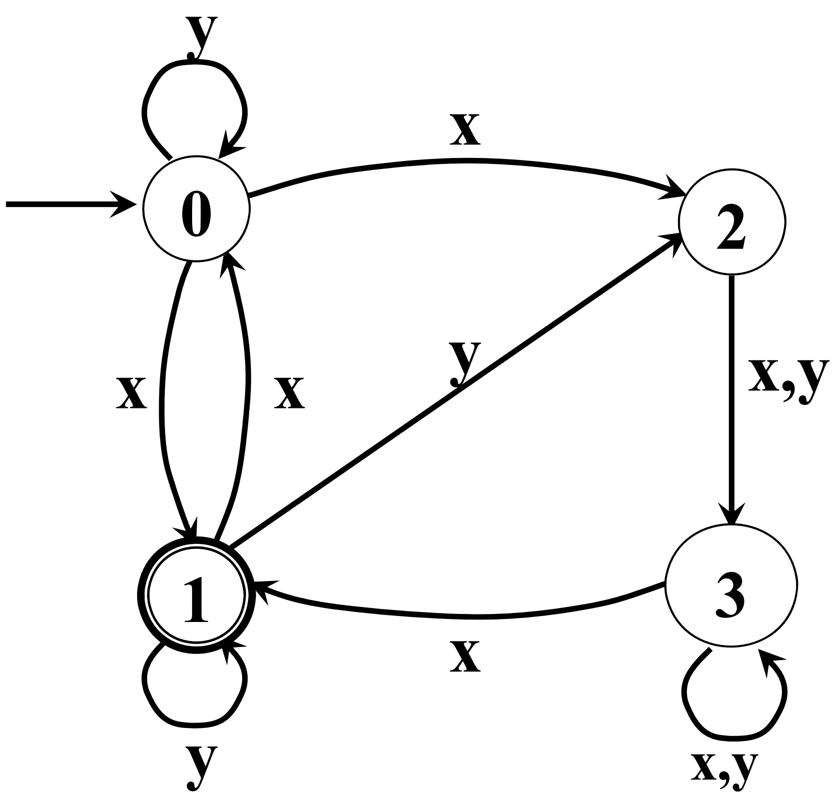
被NDFA A识别的符号串集合，记为L(A)。

NDFA举例

NDFA=({0,1,2,3}，{x,y}，f，{0}，{1})

问：xy是否可被NDFA所接受？

xyx呢？



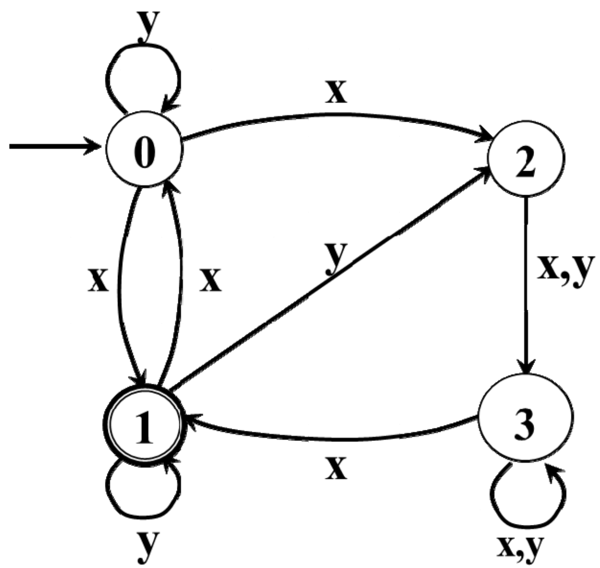
3.2 NDFA到DFA的转换

☞ 为什么要进行从NDFA到DFA的转换?

☞ 为什么要引入NDFA?

NDFA真的比DFA强大吗

- DFA是NDFA的特例。
- 对每个NDFA N一定存在一个DFA M，使得L(M)=L(N)。即对任意的NDFA N存在与之等价的DFA M。但这种DFA M可能不唯一。



3.2 NDFA到DFA的转换

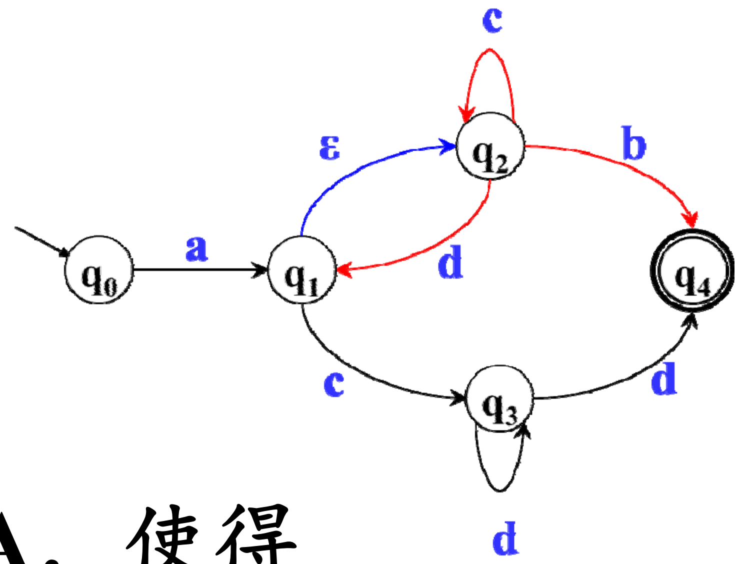
NDFA到DFA的转换：

- 确定化 —— 子集法、造表法
- **DFA**最小化 —— 构造状态集合的划分

3.2 NDFA到DFA的转换——子集法

【定义】ε自动机　　εNDFA 为（$Q, \Sigma \cup \{\varepsilon\}, t, Q_0, F$）

自动机的弧上允许标记ε，称此FA为ε自动机，记为εFA（εNDFA或εDFA）。



消除ε自动机中的空移：

对于εFA，总可以构造等价的FA，使得

$$L(\varepsilon FA) = L(FA)$$

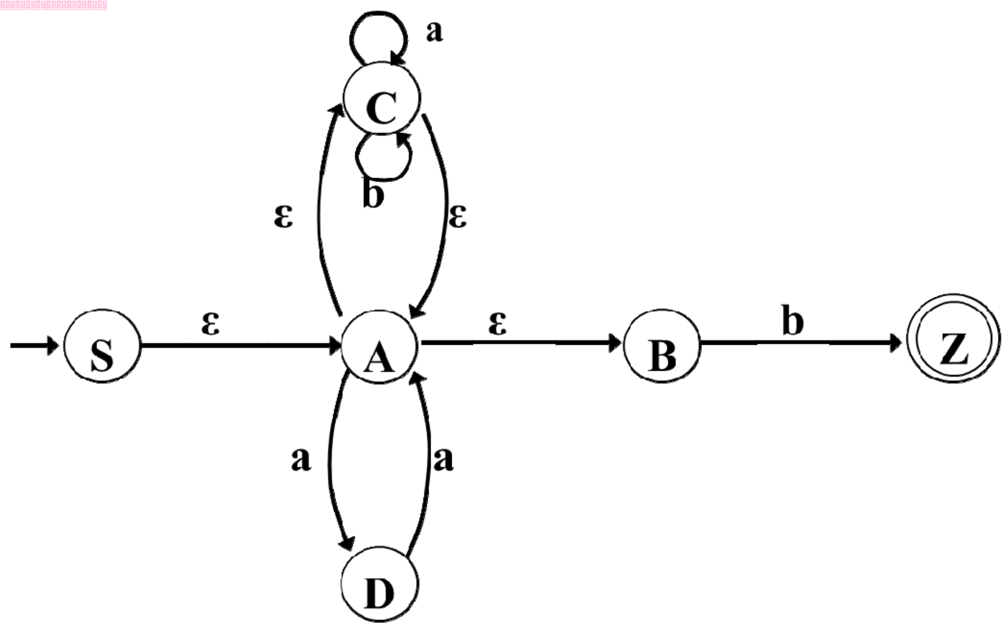
3.2 NDFA到DFA的转换——子集法

1. 子集法

(1) 空移环路的寻找和消除

(2) 消除余下的空移

(3) 利用子集法确定化



3.2 NDFA到DFA的转换——子集法

(1) 空移环路的寻找和消除

空移环路：一个从状态A开始并以A结束的空移动序列。空移环路中的所有状态是等价的。

消除方法：

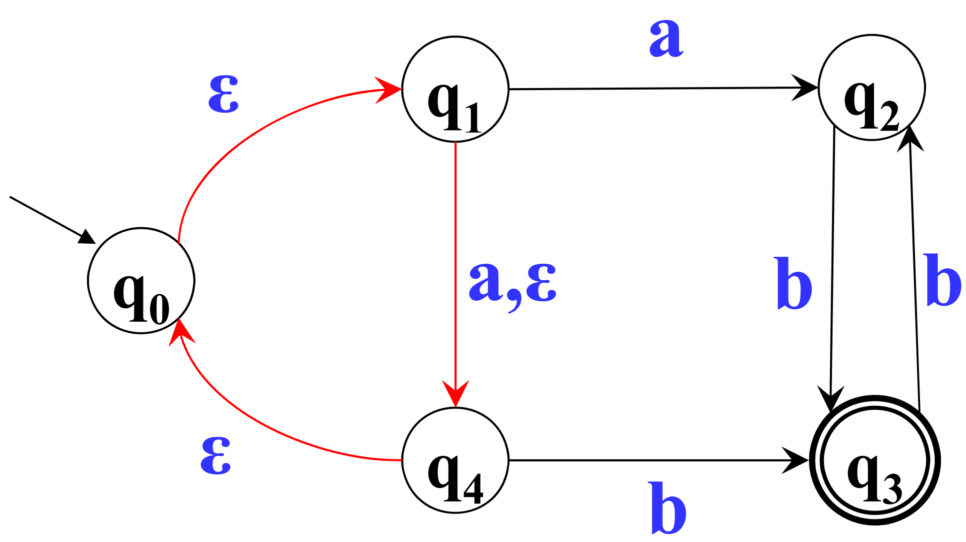
① 把空移环路上的所有结点$q_1,q_2,\ldots,q_n$合并成一个结点，取一个公共名。

② 若有$q_i \in Q_0(=q_0)\,/\,F$， $(i=1,\ldots,n)$

则合并后的新状态相应设置为初态/终态。

NDFA到DFA的转换——举例

消除空移环路：



3.2 NDFA到DFA的转换——子集法

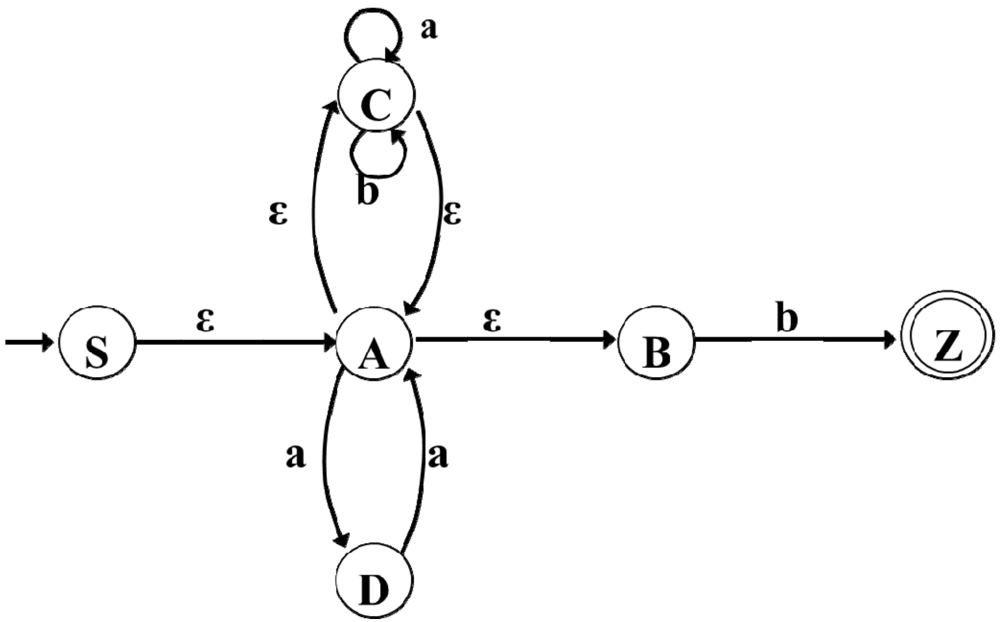
εFA ⇨ NDFA ⇨ DFA

1. 子集法

(1) 空移环路的寻找和消除

(2) 消除余下的空移

(3) 利用子集法确定化



3.2 NDFA到DFA的转换——子集法

(2) 消除空移　　εFA ⇨ NDFA

① 若 t(A,ε)={…, B, …}，则置t(A,ε)= ∅ 。

② $\forall a \in \Sigma, q \in Q$:　若$q \in t(B,a)$

把q加入$t(A,a)$

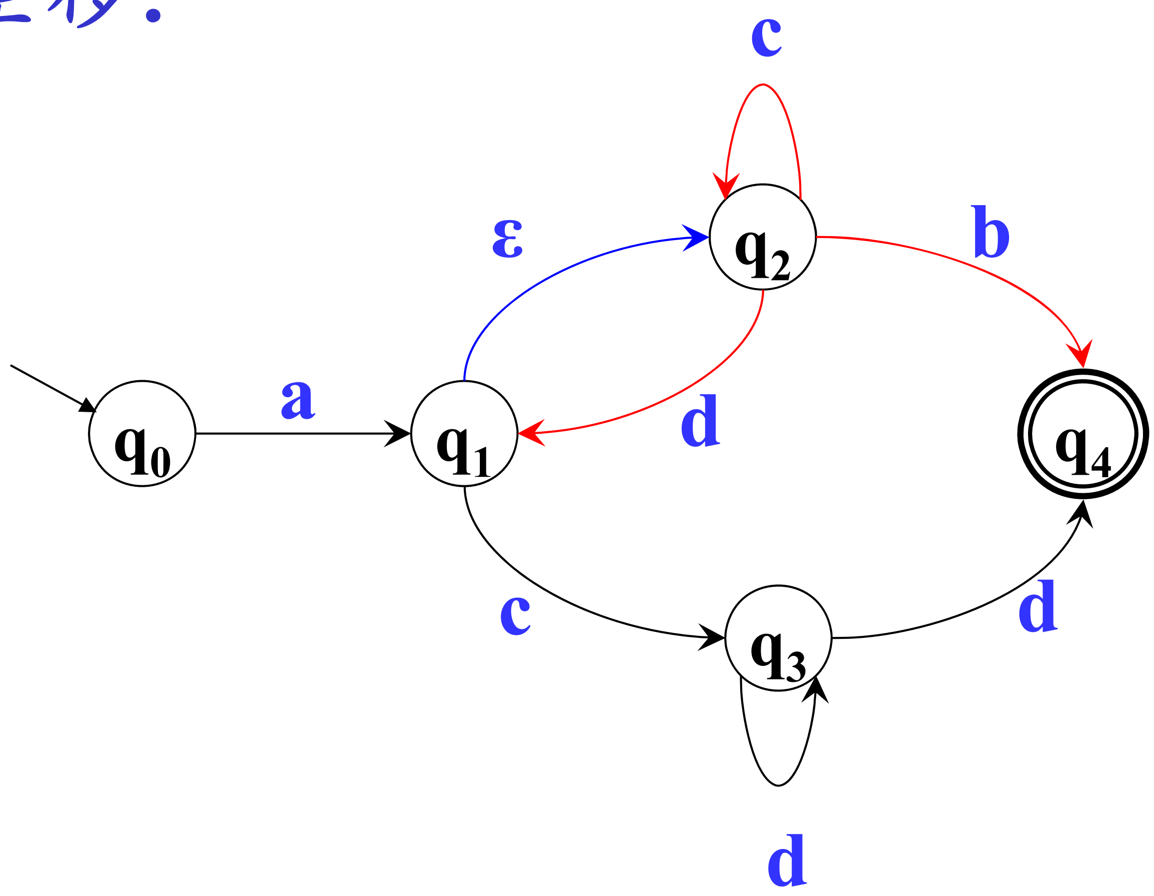
③ 若A是初态（或从初态经ε路径到达A），

则置B为初态；

若B是终态，则置A为终态。

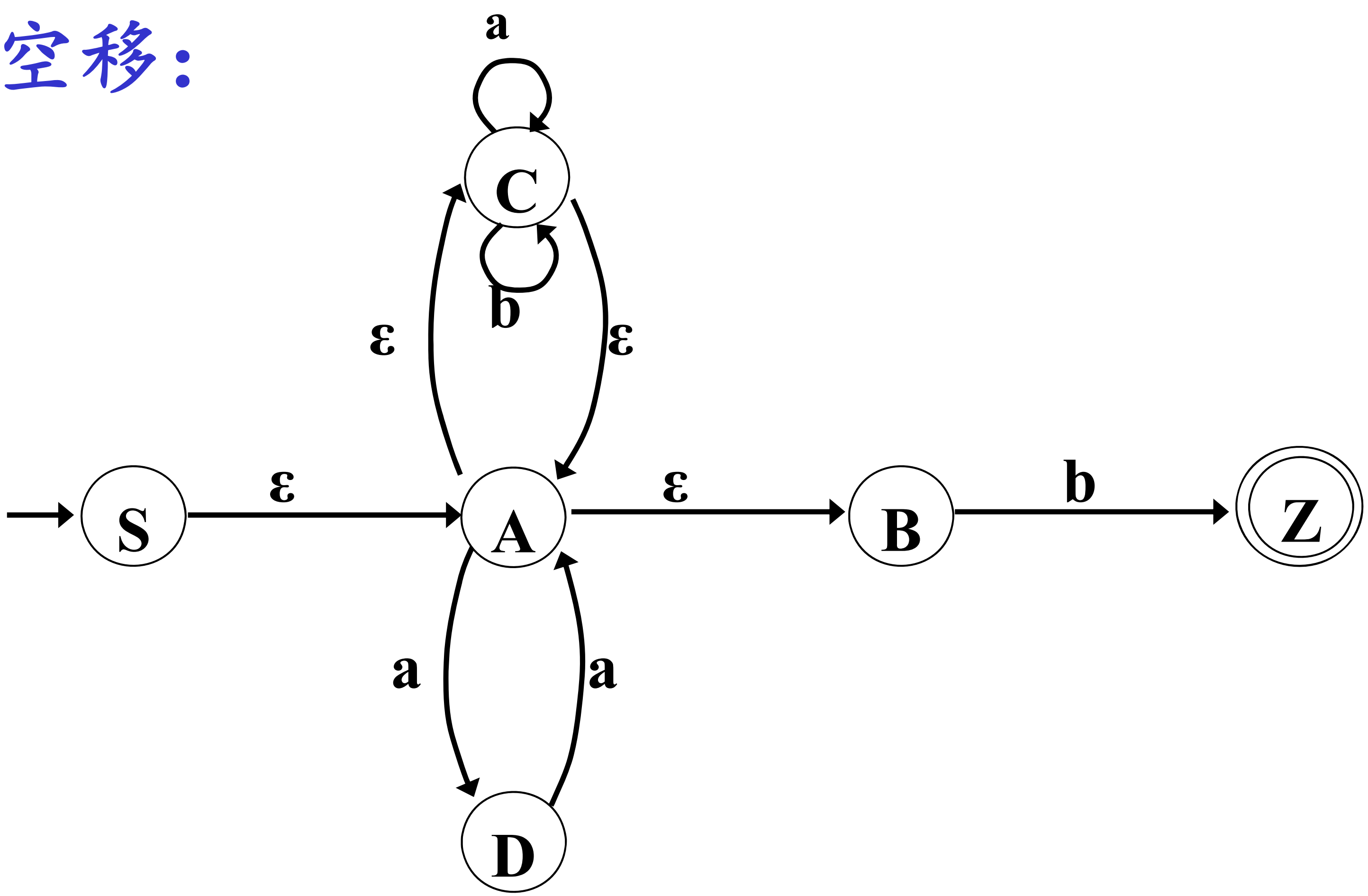
NDFA到DFA的转换——举例

消除空移:



NDFA到DFA的转换——举例

消除空移：



3.2 NDFA到DFA的转换——子集法

1. 子集法

εFA ⇨ NDFA ⇨ DFA

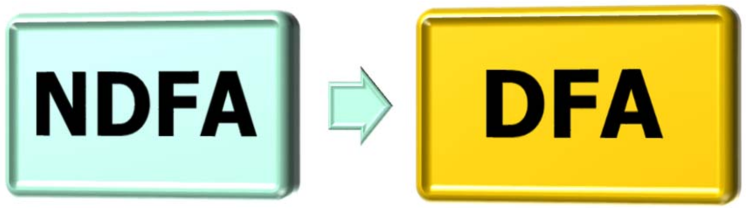
(1) 空移环路的寻找和消除

(2) 消除余下的空移

(3) 利用子集法确定化

NDFA到DFA的转换——子集法

(3) 利用子集法确定化



主要思想：

利用状态子集间确定的转换关系，进行确定化。

NDFA到DFA的转换——子集法

(3) 利用子集法确定化

NDFA ⇨ DFA

设已知的NDFA A为（Q, Σ, t, Q_0, F），

与A等价的DFA A′为（Q', Σ', t', q_0, F'）。

$$\Sigma' = \Sigma$$

$Q' = 2^Q \setminus \{\varnothing\}$，由Q 的状态子集组成

$q_0 = [s_1, s_2, \ldots, s_k]$，　$s_1, s_2, \ldots, s_k \in Q_0$

$F' = \{ [e_1,e_2,\ldots,e_p] \mid \{e_1,e_2,\ldots,e_p\} \cap F \neq \varnothing \}$

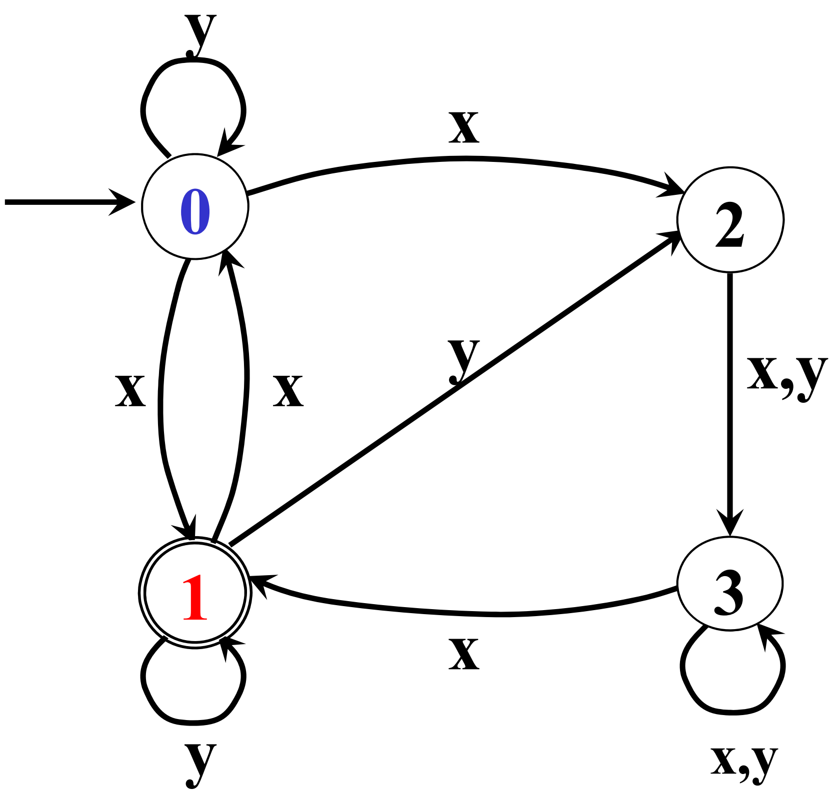
t': $t'(r',a)=q'$，其中$r'= [r_1,r_2,\ldots,r_n]$，$q'= [q_1,q_2,\ldots,q_m]$

$$\{q_1,q_2,\ldots,q_m\} = t(r_1,a) \cup t(r_2,a) \cup \ldots \cup t(r_n,a)$$

子集法举例

NDFA ⇨ DFA

DFA=(Q',{x,y}, t', [0], {[1],[0,1],[1,2],[1,3],[0,1,2],[0,1,3], [1,2,3],[0,1,2,3]})



t '([0],x)=[1,2] t '([0],y)=[0]

t '([1],x)=[0] t '([1],y)=[1,2]

t '([2],x)=[3] t '([2],y)=[3]

t '([3],x)=[1,3] t '([3],y)=[3]

t '([0,1],x)=[0,1,2] t '([0,1],y)=[0,1,2]

t '([0,2],x)=[1,2,3] t '([0,2],y)=[0,3]

t '([0,3],x)=[1,2,3] t '([0,3],y)=[0,3]

t '([1,2],x)=[0,3] t '([1,2],y)=[1,2,3]

子集法举例

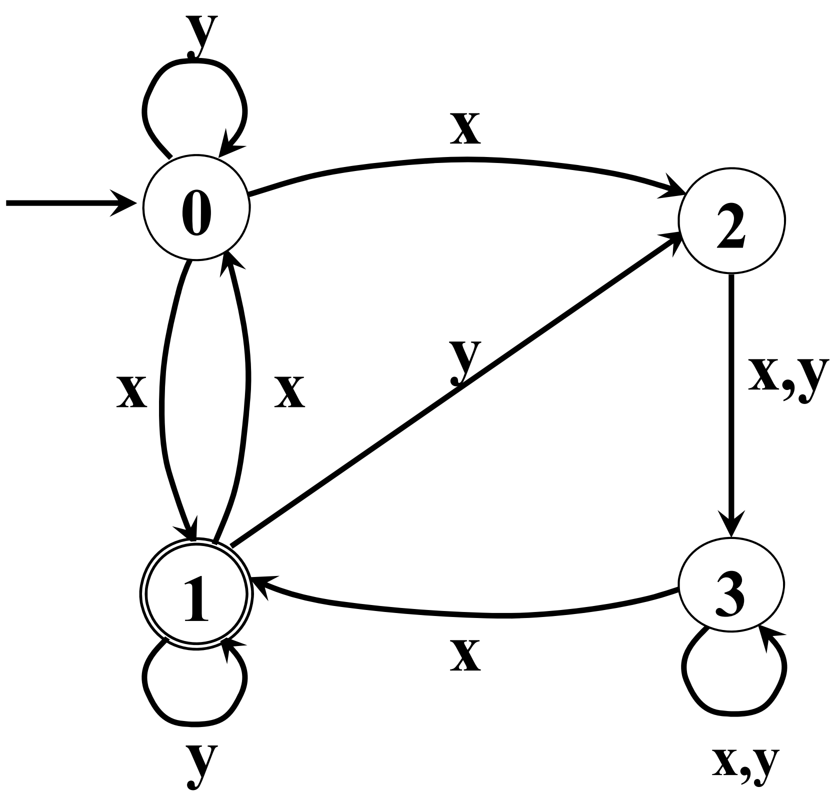
☞ 为什么可以用子集法进行NDFA的确定化？

DFA=(Q',{x,y}, t', [0], {[1],[0,1],[1,2],[1,3],[0,1,2],[0,1,3], [1,2,3],[0,1,2,3]})

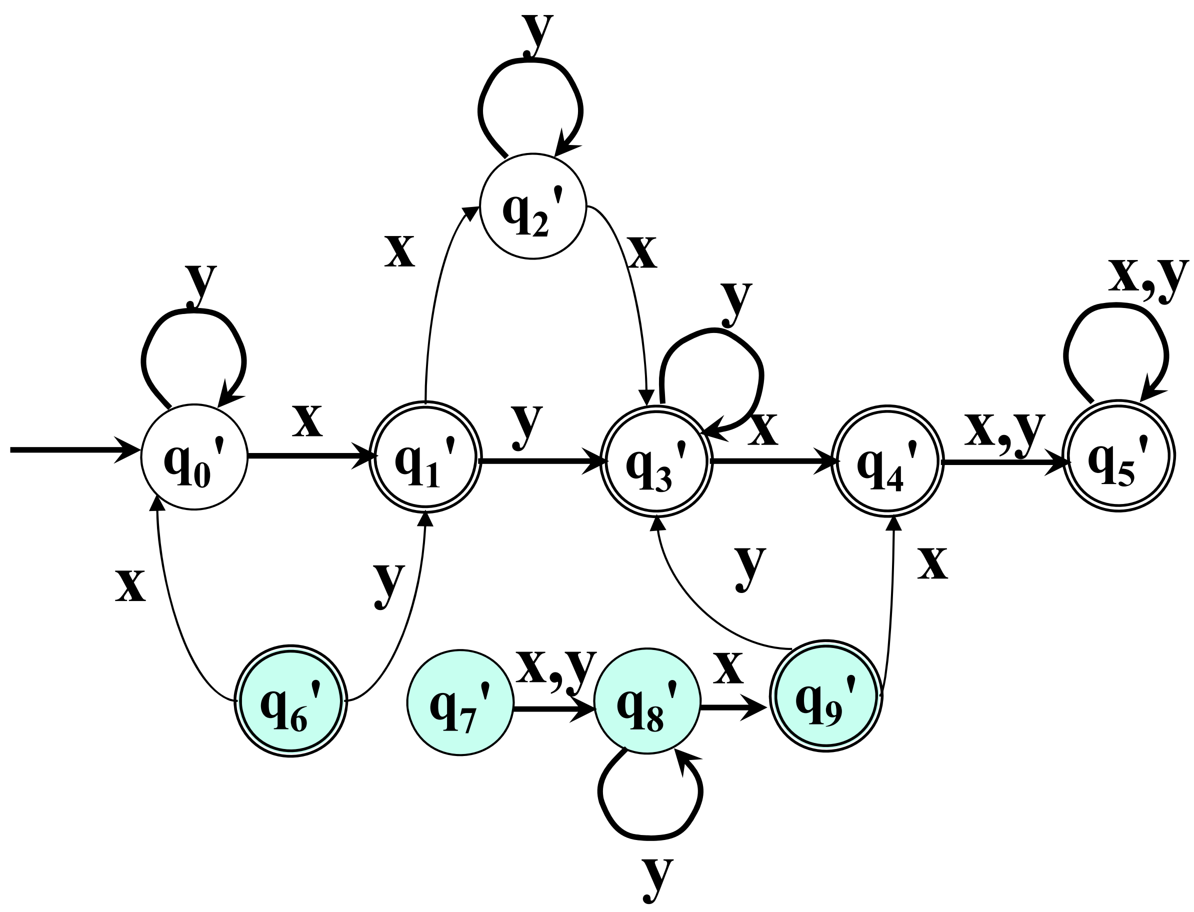
t '([0],x)=[1,2]	t '([0],y)=[0]	t '([1,3],x)=[0,1,3]	t '([1,3],y)=[1,2,3]
t '([1],x)=[0]	t '([1],y)=[1,2]	t '([2,3],x)=[1,3]	t '([2,3],y)=[3]
t '([2],x)=[3]	t '([2],y)=[3]	t ' ([0,1,2],x)=[0,1,2,3]	t '([0,1,2],y)=[0,1,2,3]
t '([3],x)=[1,3]	t '([3],y)=[3]	t '([0,1,3],x)=[0,1,2,3]	t '([0,1,3],y)=[0,1,2,3]
t '([0,1],x)=[0,1,2]	t '([0,1],y)=[0,1,2]	t ' ([0,2,3],x)=[1,2,3]	t '([0,2,3],y)=[0,3]
t '([0,2],x)=[1,2,3]	t '([0,2],y)=[0,3]	t ' ([1,2,3],x)=[0,1,3]	t '([1,2,3],y)=[1,2,3]
t '([0,3],x)=[1,2,3]	t '([0,3],y)=[0,3]	t '([0,1,2,3],x)=[0,1,2,3]	t '([0,1,2,3],y)=[0,1,2,3]
t '([1,2],x)=[0,3]	t '([1,2],y)=[1,2,3]		

子集法举例

NDFA ⇨ DFA



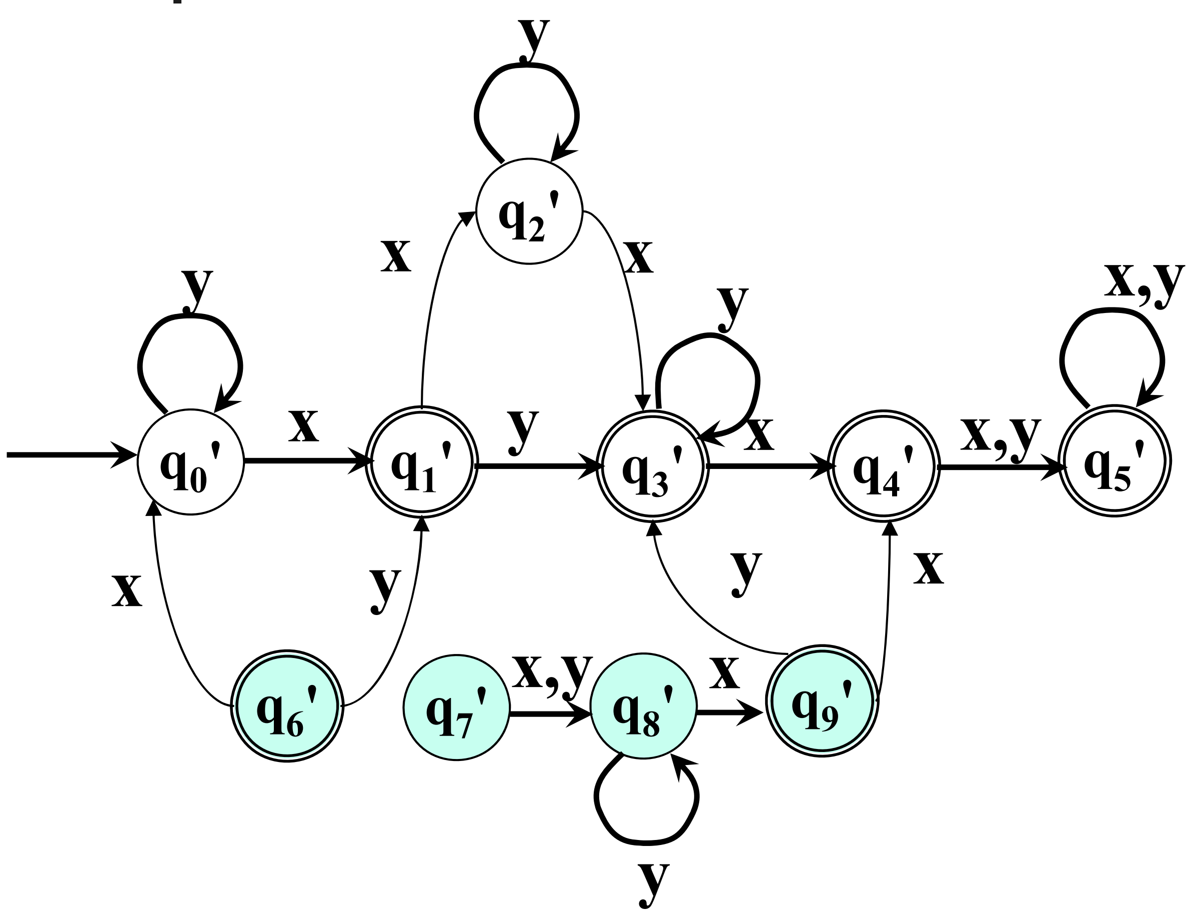
NDFA的状态转换图



DFA的状态转换图[部分]

子集法的局限

NDFA ⇨ DFA



DFA的状态转换图[部分]

子集法的局限性：

- 状态数太多
- 存在不可达状态

➡ 造表法

NDFA到DFA的转换——造表法

利用造表法确定化

NDFA ⇨ DFA

主要思想：

为避免不可达状态，从初始状态出发，计算t′，依次构造其后继状态，进行确定化。

$$I\text{子集} \xrightarrow{a} I_a\text{子集}$$

$$I_a = t'(I, a)$$

3.2 NDFA到DFA的转换——造表法

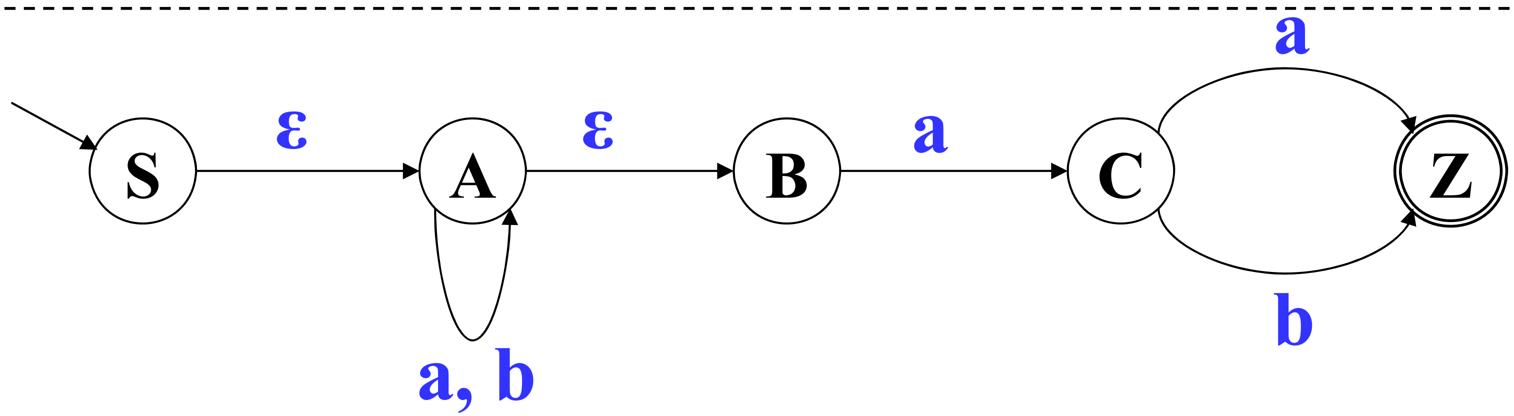
- 求某个子集I在a下的后继状态（子集）I_a：

即从I中状态出发，经过一条a弧（跳过a弧前、越过a弧后的任意条ε弧）到达的状态集合（子集）。

① 状态子集I的ε闭包

② I_a子集

$$I_a = t'(I, a)$$



I={S}，求I_a = ?

3.2 NDFA到DFA的转换——造表法

① 状态子集I的ε闭包

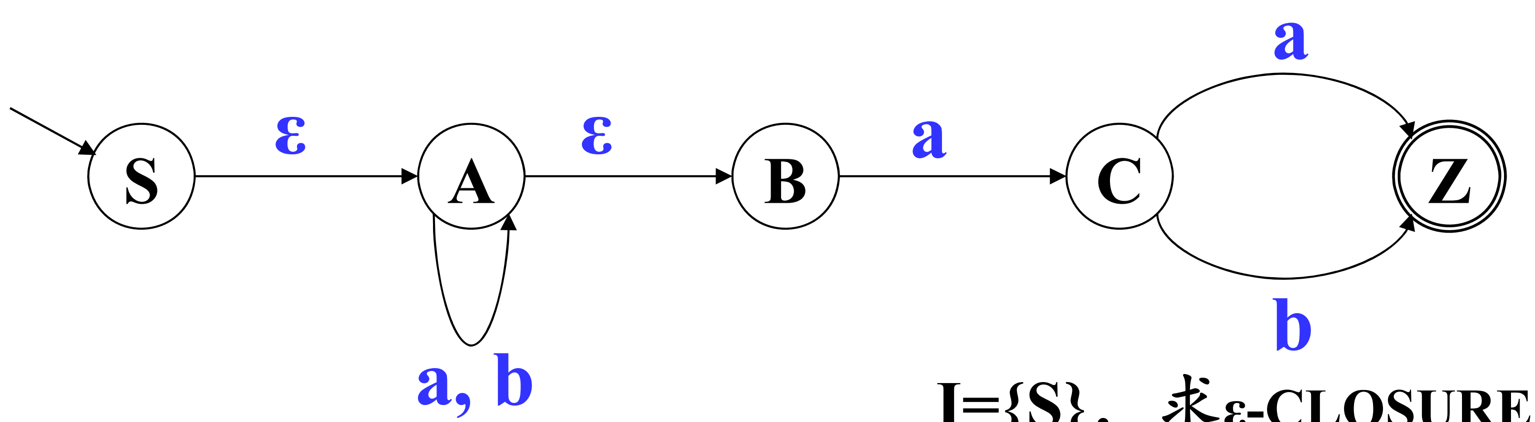
假设I是状态集合Q的一个子集。

定义I的ε闭包（ε-CLOSURE（I））为：

ⅰ.如果状态q∈I，则q∈ε-CLOSURE（I）；

ⅱ.如果状态q∈I，则q′∈ε-CLOSURE（I）。

其中q′为由状态q出发，经任意条ε弧能到达的Q中的状态。



I={S}，求ε-CLOSURE（I）=？

3.2 NDFA到DFA的转换——造表法

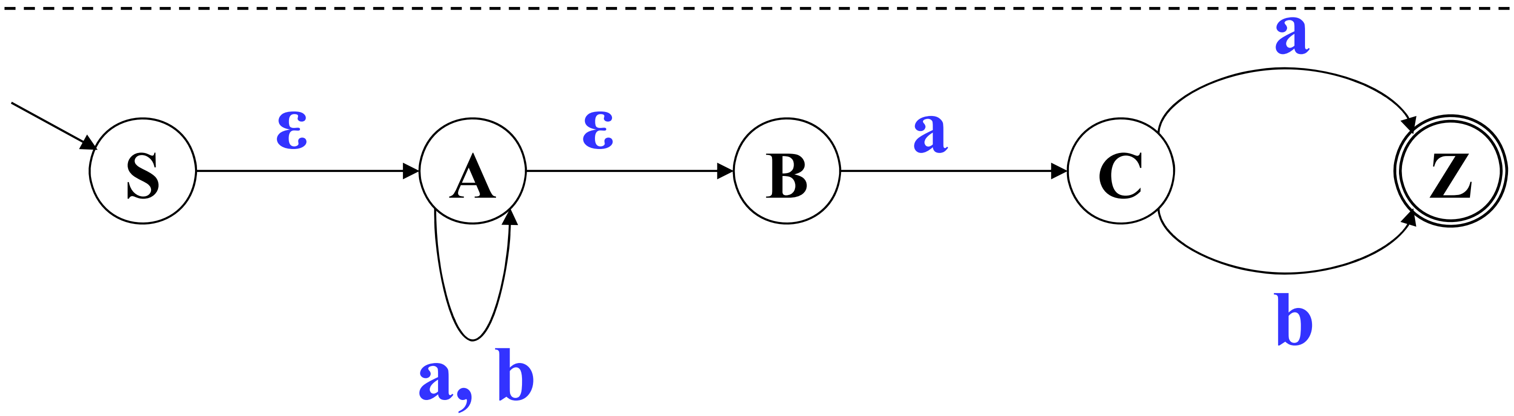
②I_a子集

$I_a = t'(I, a)$

设I是状态集Q的一个子集。

由I中的状态出发，经历一条a弧（跳过a弧前的任意条ε弧）可到达的状态的集合称为J，则

I_a=ε-CLOSURE（J）



I={S, A}，求I_a＝?

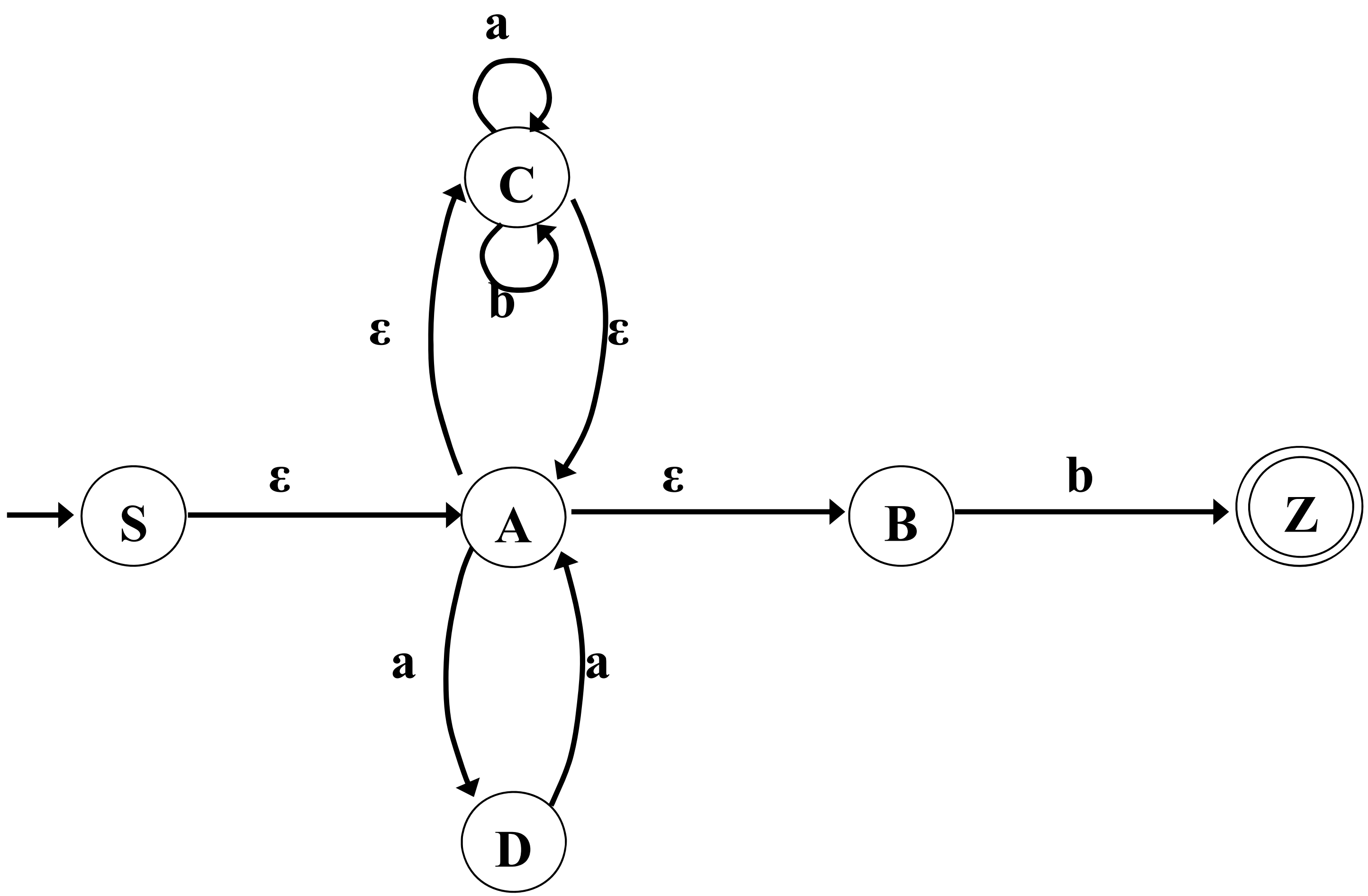
NDFA到DFA的转换——造表法

利用造表法确定化

1. 将I=ε-closure(Q_0)作为表的第1行第1列，然后求其I_a和I_b，填入第2，3列；
2. 将未出现在第1列的I_a和I_b依次填入下一行作为I值，并求其I_a和I_b，填入相应行的第2，3列；
3. 重复2，直至所有2，3列元素全部在第1列出现过为止；
4. 最终表即是状态转换表。

状态：	第1列元素
输入符号：	I_a和I_b看作a和b
状态转换函数：	表中相应元素
初态：	第1行第1列元素
终态：	含原终态的元素

NDFA到DFA的转换——举例



NDFA到DFA的转换——造表法

造表法的特点：

- 简单有效
- 不存在不可达状态
- 状态数比子集法大大减少

最少?

→ DFA化简(最小化)

3.2 NDFA到DFA的转换

NDFA到DFA的转换：

- 确定化 —— 子集法、造表法
- DFA最小化 —— 构造状态集合的划分

3.2 NDFA到DFA的转换——DFA化简

寻找一个状态数比M更少的DFA M′，使得M和M′所识别的字符串相同，即L(M)=L(M′)。（M′唯一的）

条件：接受的语言相同（等价的DFA）

主要思想：

- 合并等价状态
- 删除无关状态

3.2 NDFA到DFA的转换——DFA化简

等价状态：如果从DFA 的某个状态q_1出发能识别某一字符串x而停止于终态，那么从q_2出发也能识别字符串x而停止于终态；反之亦然。则称状态q_1 和q_2 是等价的。

如果q_1 和q_2 不等价，则说q_1 和q_2 是可区分的。

最小化算法（划分法）：

DFA最小化的关键在于把它的状态集分成一些两两互不相交的子集，使得任何两个不同的子集中的状态都是可区分的，而同一子集中的任何两个状态都是等价的。

3.2 NDFA到DFA的转换——DFA化简

步骤一　构造状态集的划分

- 终止状态集/非终止状态集
- 对每一个子集进行再分解（属于同一子集的任意状态q_1和q_2，对于任何$a \in \Sigma$，$t(q_1,a)$与$t(q_2,a)$属于同一子集。）

步骤二　取每一组中的一个状态作代表，合并等价状态

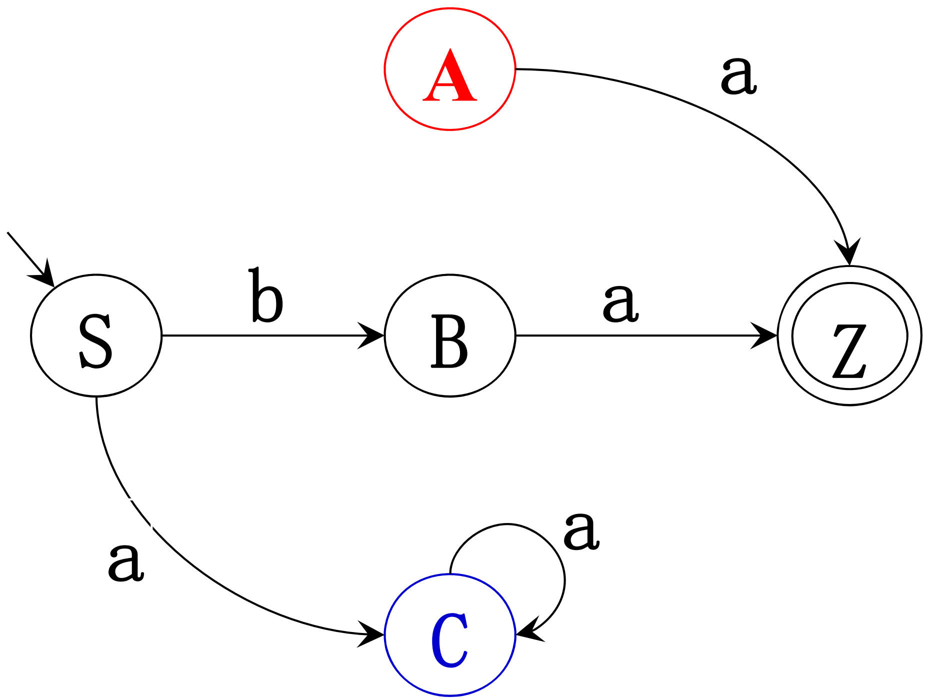
步骤三　删去无关状态

不可达状态

死状态

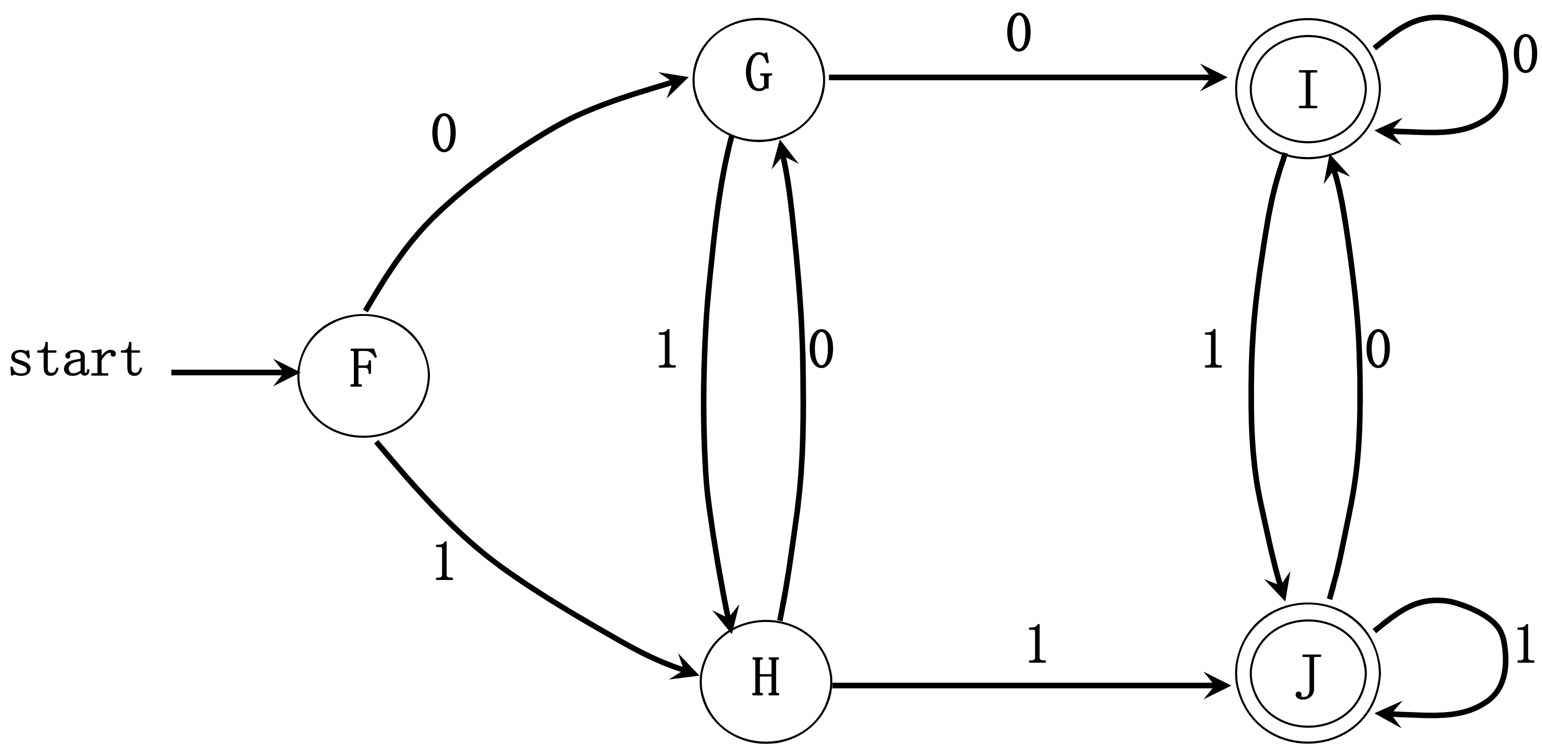
DFA化简举例

[例] 删去无关状态

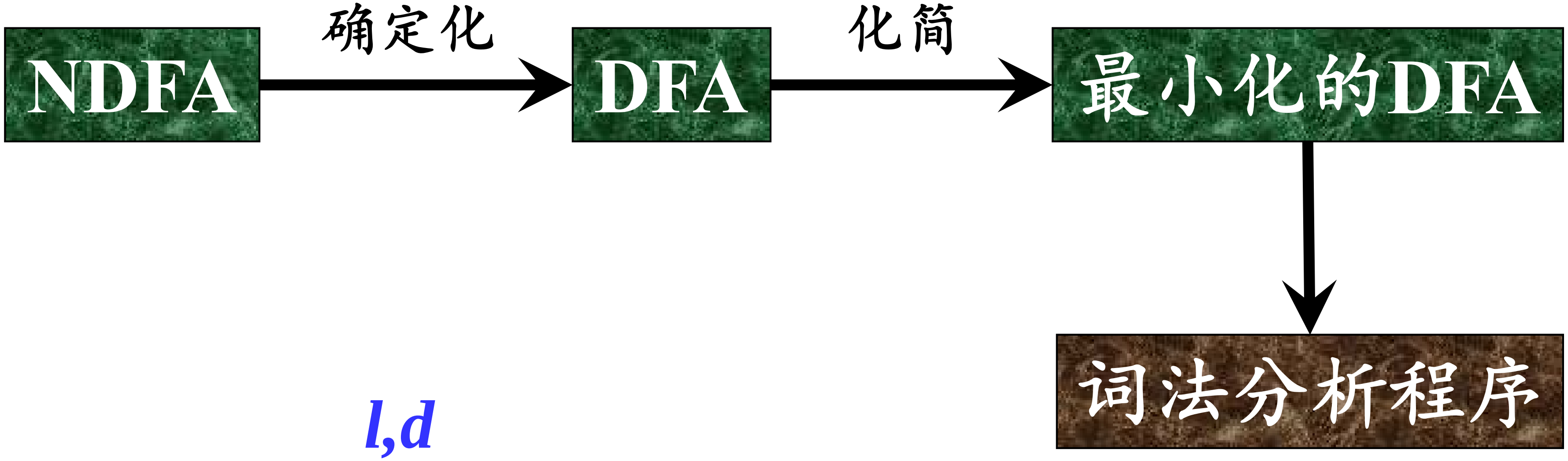


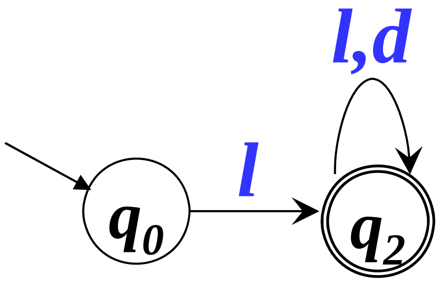
DFA化简举例

[例] 将下列DFA最小化。



从化简后的DFA到程序表示

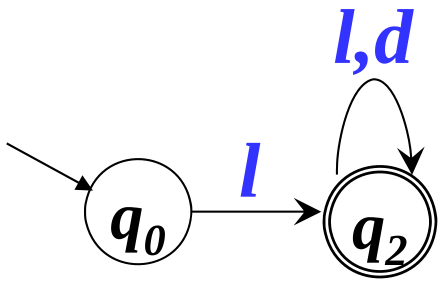


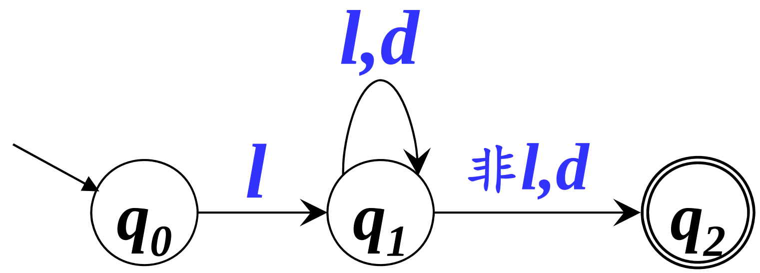


标识符DFA

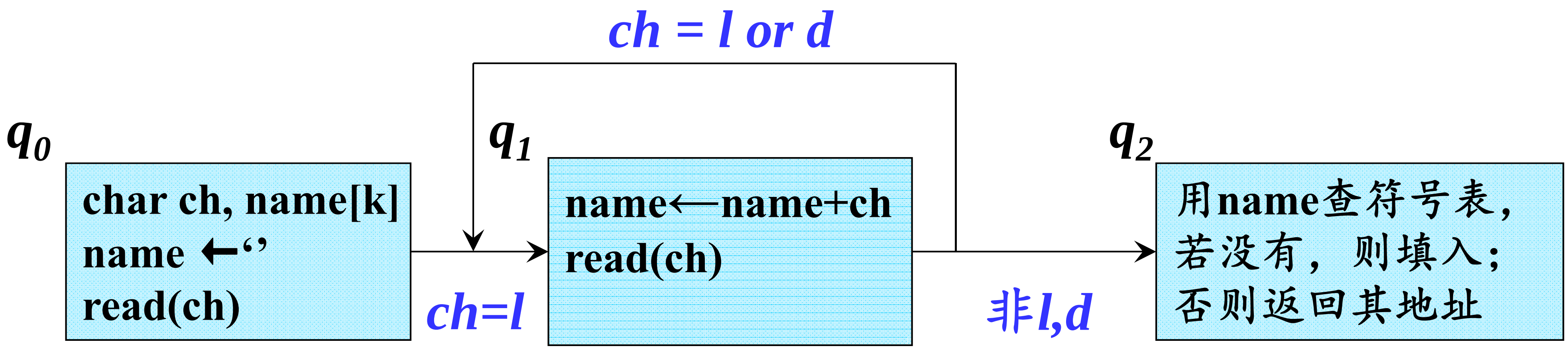
Where are we?
Why are we doing this?

从化简后的DFA到程序表示



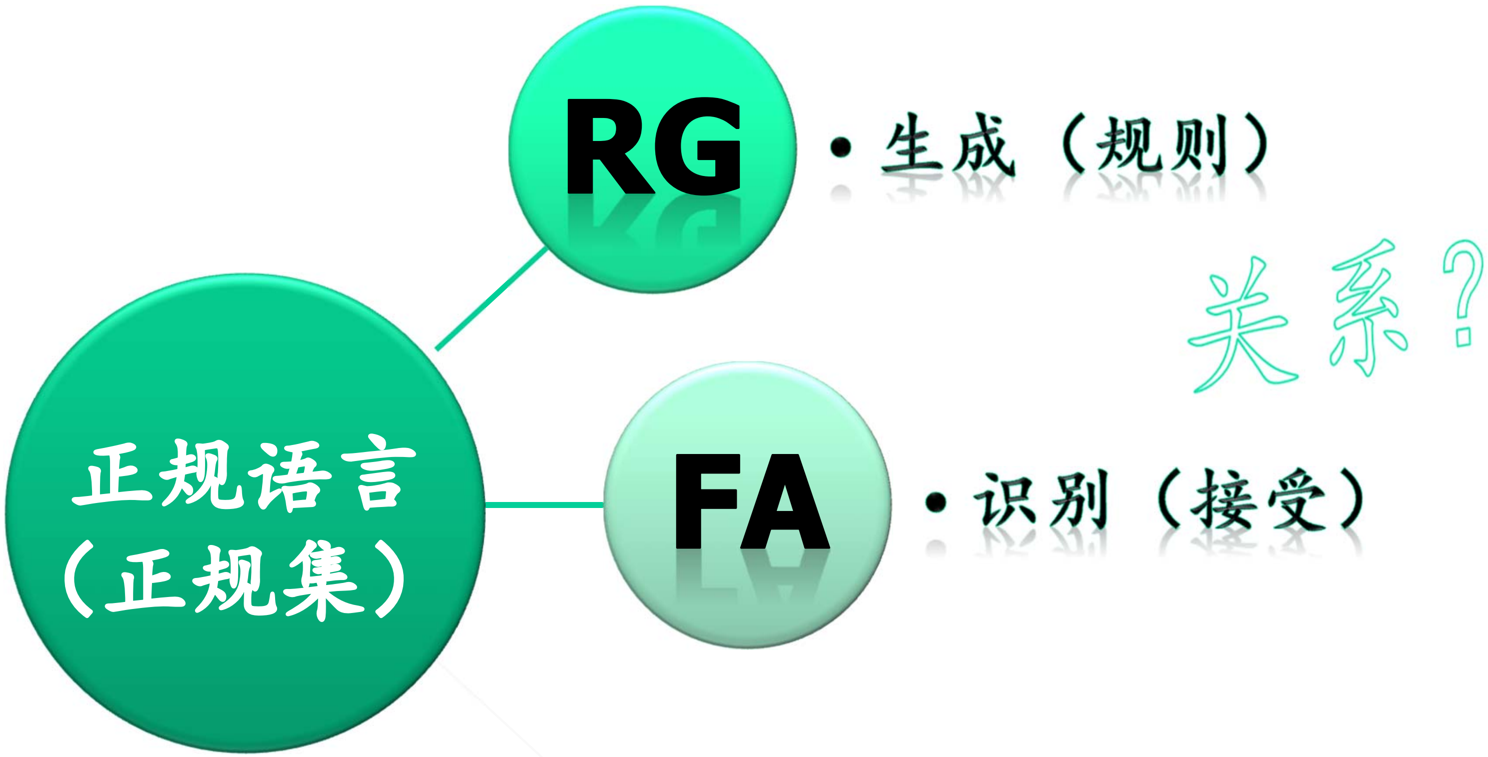


标识符DFA



识别标识符的程序流程图

3.3 正规文法与FA



3.3 正规文法与FA

[定理1] RG→FA

由正规文法G[S]可直接构造一个与之等价的FA A，使得L(G)=L(A)。

[定理2] FA→RG

由有穷自动机FA A可直接构造一个与之等价的正规文法G，使得L(G)=L(A)。

3.3 正规文法与FA

1. RG→FA的构造：

- 令G的终结符号集为A的字母表；
- G的非终结符号作为A的状态，G的开始符号为A的开始状态；
- 增加一个终止状态Z（$Z \notin V_N$）；
- 形如$U \rightarrow a$的规则，引一条从状态U到终止状态Z的标记为a的 弧；
- 形如$U \rightarrow aW$的规则，引一条从状态U到W的a弧。

正规文法⇒FA

【例1】给定文法G[Z]：

Z→0U|1V

U→1Z|1

V→0Z|0

试给出等价的FA A。

【例2】给定文法G[S]：

S→aS|aB

B→bB|bA

A→cA|c

试给出等价的FA A。

FA⇒正规文法

2. FA→RG的构造：

①自动机A中的每一个状态均作为G的非终结符号，其中A的开始状态作为G的开始符号，A的输入字母表中的所有符号作为G的终结符号；

②对A中$V \in t(U,a)$的映射，构造G的产生式U∷=aV；

若$V \in F$，则构造G的产生式 U::=a；

③若A中$q_0 \in F$，则构造G的产生式S∷=ε。

FA⇒正规文法

2. FA→RG的构造：

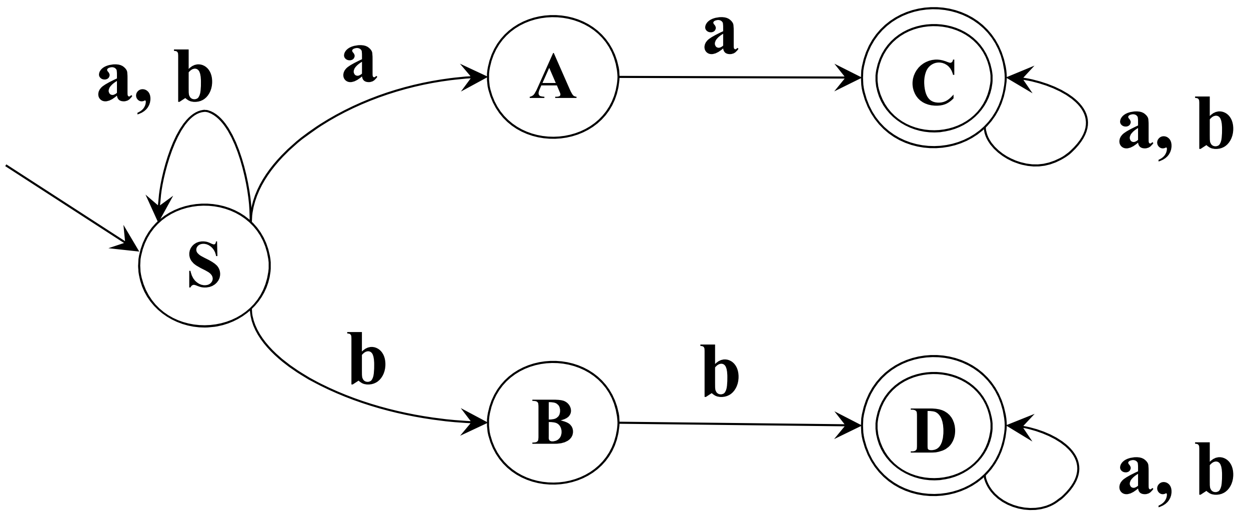
①自动机A中的每一个状态均作为G的非终结符号，其中A的开始状态作为G的开始符号，A的输入字母表中的所有符号作为G的终结符号；

②对A中$V \in t(U,a)$的映射，构造G的产生式U∷=aV；

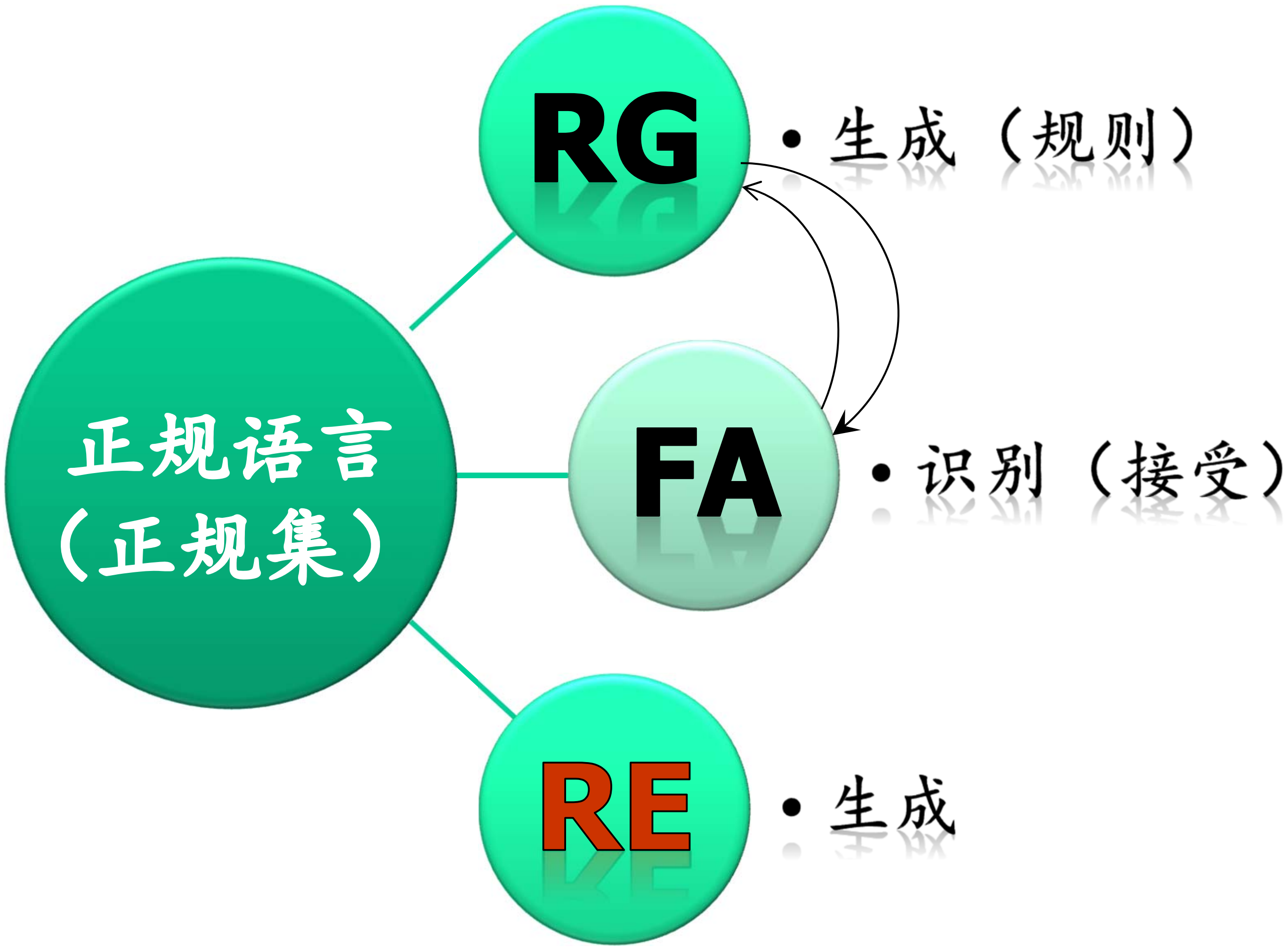
③对A中终止状态Z，构造G的形如Z∷=ε的产生式。

FA ⇒ RG举例

[例] 为下列NDFA A构造等价的正规文法。（P46，3.9）



3.3 正规文法与FA



3.4 正规表达式RE与FA

字母表Σ上的正规表达式RE e，描述的语言称为正规集L(e)。

(1)ε、∅是Σ上的RE，相应的正规集分别是{ε}、∅。

(2)$\forall a \in \Sigma$，a是Σ上的RE，相应的正规集是{a}；

(3)设e_1与e_2是Σ上的RE，分别描述正规集$L(e_1)$、$L(e_2)$，则

①(e_1)也是Σ上的RE，相应正规集为$L((e_1))=L(e_1)$；

②e_1e_2也是Σ上的RE，相应正规集为$L(e_1e_2)=L(e_1)L(e_2)$；

③$e_1|e_2$也是Σ上的RE，相应的正规集为$L(e_1|e_2)=L(e_1)\cup L(e_2)$；

④e_1^*也是Σ上的RE，相应的正规集为$L(e_1^*)=(L(e_1))^*$。

仅由有限次(3)所描述运算而得到的表达式才是Σ上的正规表达式。

正规表达式举例

（1） 0^*10^*

（2） $\Sigma^* 1\Sigma^*$

（3） $\Sigma^* 001\Sigma^*$

（4） $(\Sigma\Sigma)^*$

（5） $(\Sigma\Sigma\Sigma)^*$

（6） $0\Sigma^*0 \mid 1\Sigma^*1 \mid 0 \mid 1$

（7） $(0 \mid \varepsilon)1^* = 01^* \mid 1^*$

（8） $\Phi^*=\{\varepsilon\}$

3.4 RE与FA——正规表达式等价

【定义】正规表达式等价

设e_1，e_2均为Σ上的正规表达式，

若 $L(e_1)=L(e_2)$

则称e_1与e_2等价，记为：$e_1=e_2$。

[例] 试证：$b(ab)^* = (ba)^*b$

3.4 RE与FA——RE的性质

设A、B、C均为正规表达式，则有下列等价关系成立：

(1) $A|B=B|A$

(2) $A|(B|C)=(A|B)|C$

(3) $A(BC)=(AB)C$

(4) $A(B|C)=AB|AC$ $(B|C)A=BA|CA$

(5) $\varepsilon A=A\varepsilon=A$

(6) $(A^*)^*=A^*$

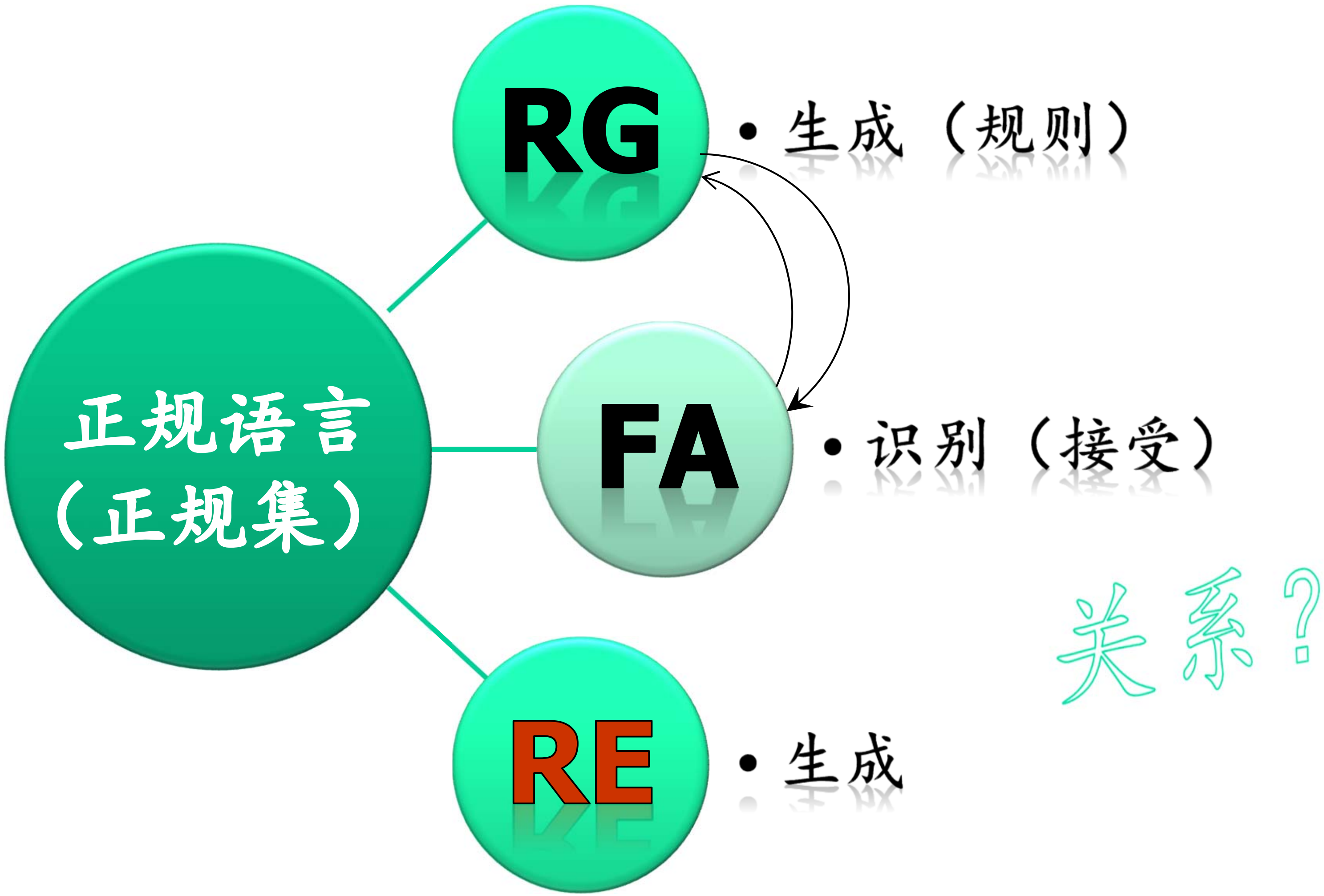
(7) $A^*=\varepsilon|AA^*$

(8) $(AB)^*A=A(BA)^*$

(9) $(A|B)^*=(A^*B^*)^*=(A^*|B^*)^*$

(10) $A=b|aA$ 当且仅当 $A=a^*b$ (11) $A=b|Aa$ 当且仅当 $A=ba^*$

RG、FA和RE的关系



3.4 RE与FA——关系

[定理1] **RE→FA**

对于字母表Σ上的任意正规表达式e，一定可以构造一个输入字母表Σ上的NDFA A，使得L(A)=L(e)。

[定理2] **FA→RE**

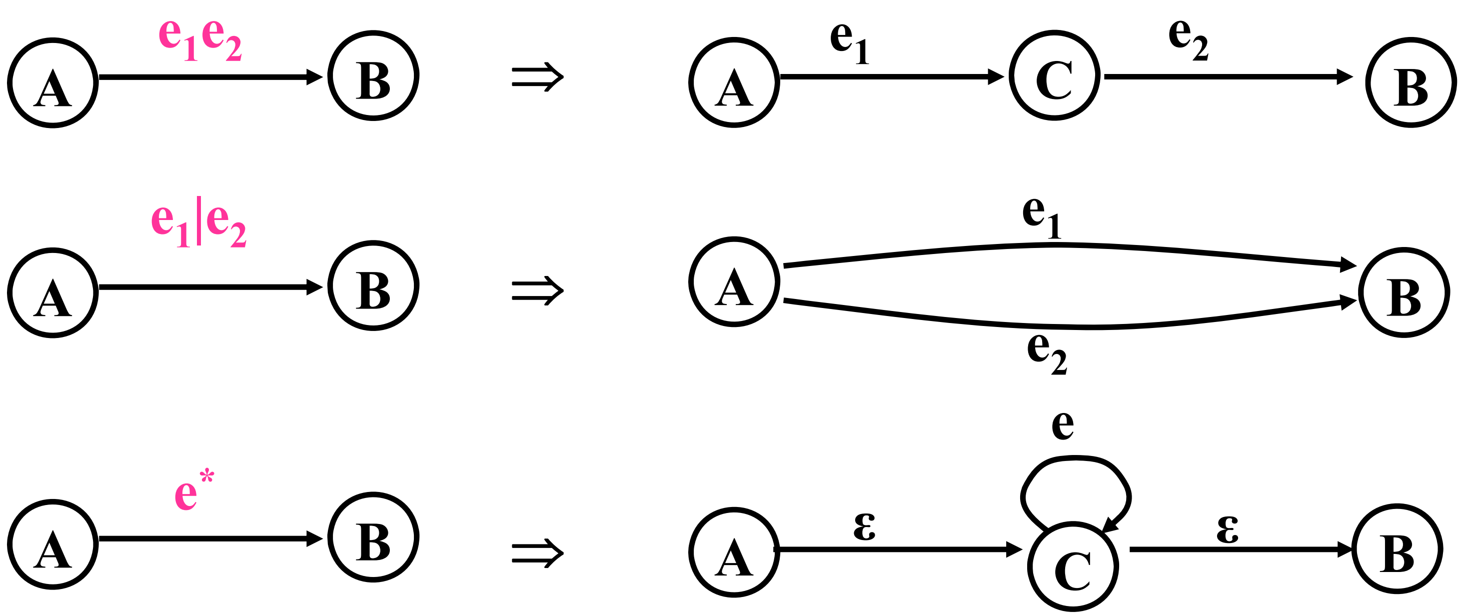
由有穷自动机FA A所识别的语言L(A)，可以用Σ上的RE e来表示，使得L(A)=L(e)。

3.4 RE与FA——RE⇒FA

1. RE→FA

(1)构造广义NFA：S是惟一开始状态，Z是惟一终止状态。弧标记e。

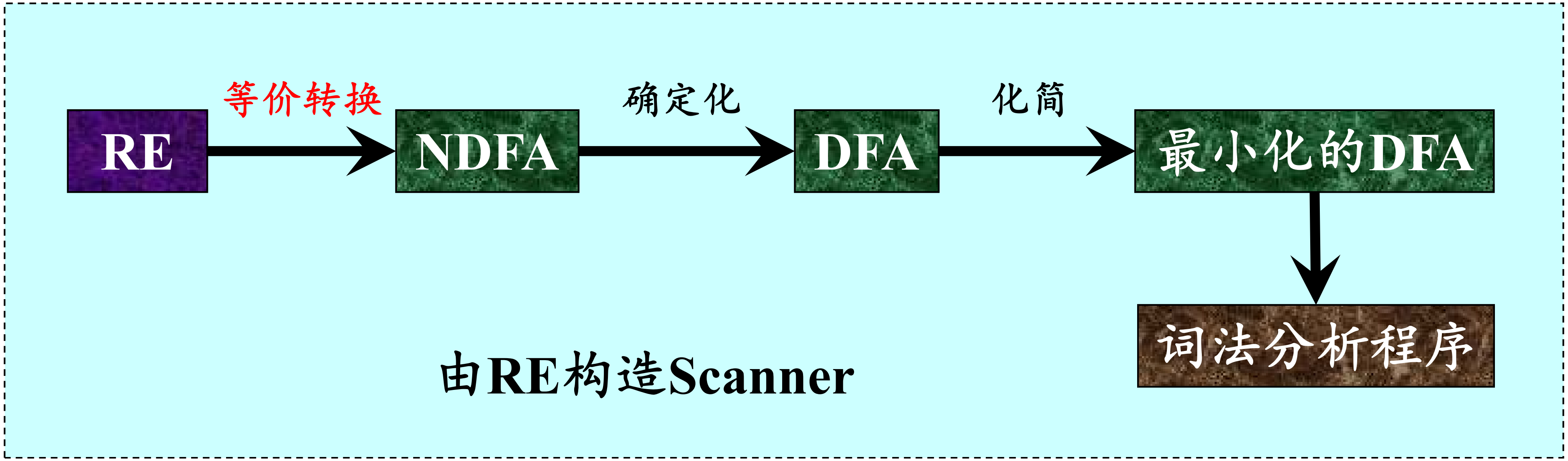
(2)根据分解规则分解e，得到与e对应的NFA。



RE⇒FA举例

[例] 构造正规表达式 xy*|yx*y|xyx 相应的NDFA。

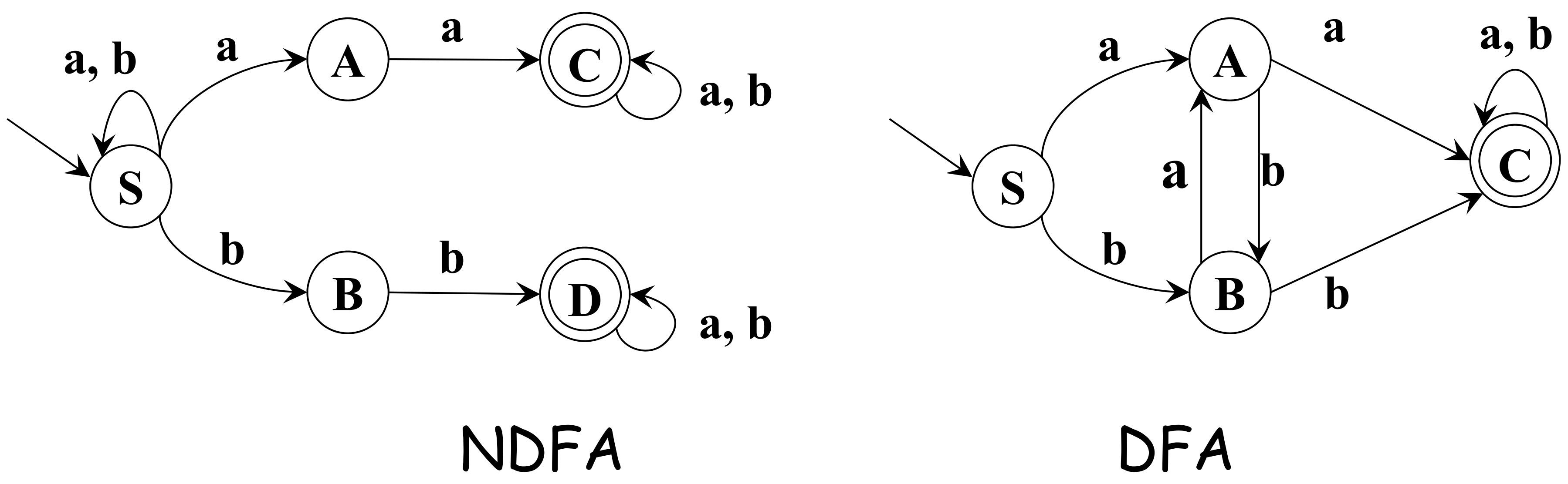
☞ 为什么要引入NDFA？



RE⇒FA举例

[例] 构造正规表达式 xy*|yx*y|xyx 相应的NDFA。

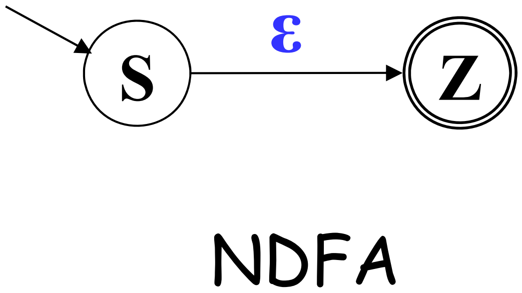
☞ 为什么要引入NDFA？

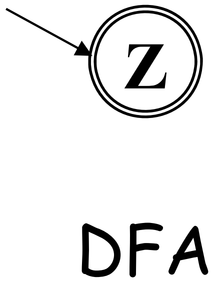


RE⇒FA举例

[例] 构造正规表达式 xy*|yx*y|xyx 相应的NDFA。

☞ 为什么要引入NDFA？



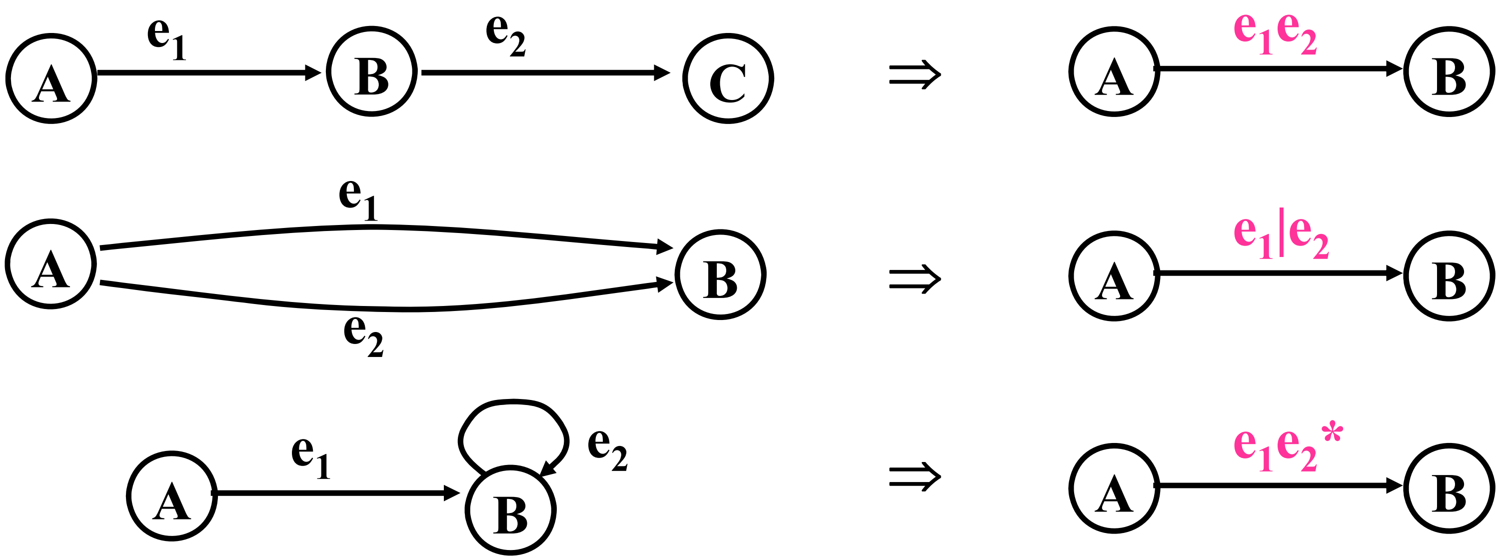


3.4 RE与FA——FA⇒RE

2. FA→RE

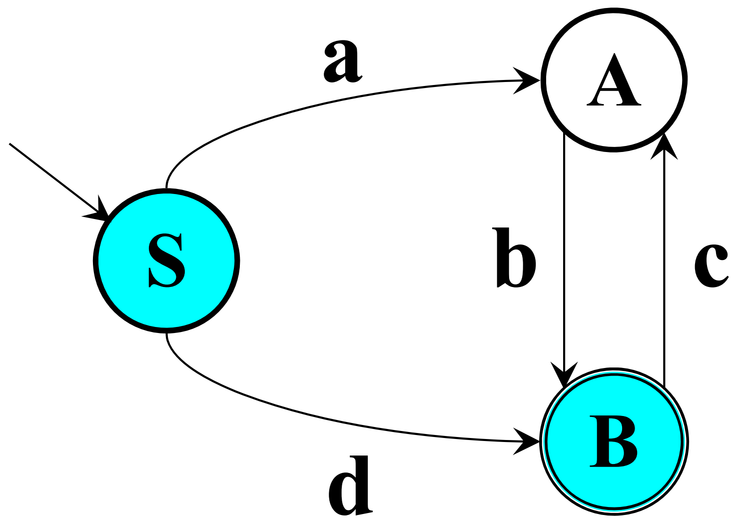
(1)在FA A的状态图中增加两个结点：S是惟一开始状态，从S向原开始状态连ε弧；Z是惟一终止状态，从原终止状态向Z连ε弧。

(2)利用下列替换规则逐步消去状态图中的结点和弧，直至仅剩下S到Z的一条弧为止，则该弧上的标记即为RE e。

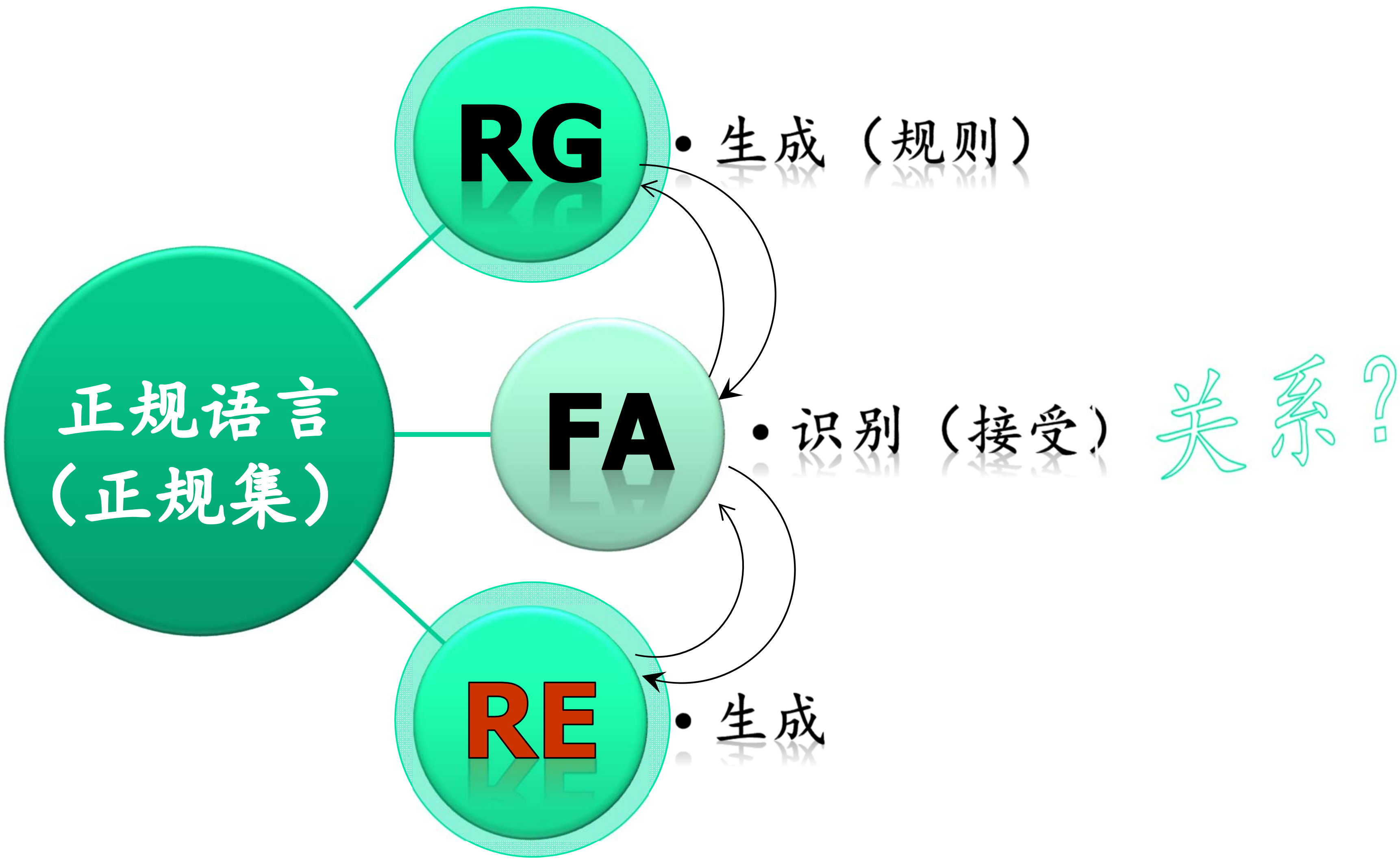


FA⇒RE举例

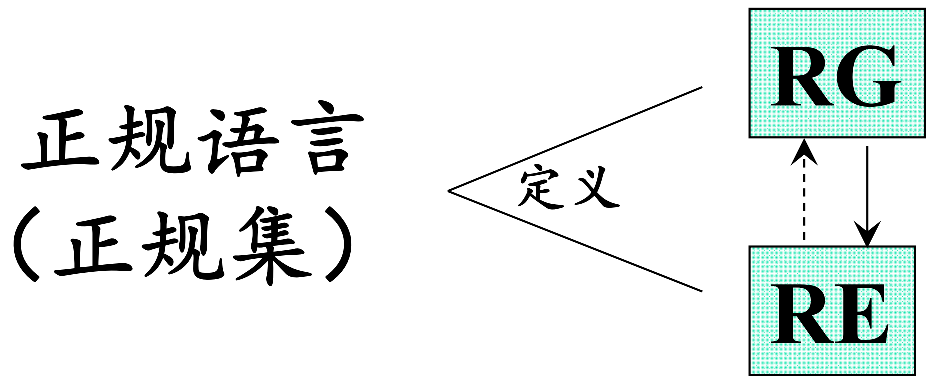
[例] 求与下列FA等价的正规表达式。



RG、FA和RE的关系



3.5 正规文法到正规表达式



[定理1] RG→RE
对任何一个正规文法G，都存在一个正规表达式e，使得L(e)=L(G)。

[定理2] RE→RG
对任何一个正规表达式e，都存在一个正规文法G，使得L(G)=L(e)。

3.5 正规文法到正规表达式

1. RG→RE

对任何一个正规文法G，都存在一个等价的正规表达式e，使得L(e)=L(G)。

将正规文法拓广：

产生式的形式为 $U \rightarrow \alpha V$ 或 $U \rightarrow \alpha$， α为可空字符串，

即右线性文法，容易改写成RG。

$RG \Rightarrow RE$

1. RG→RE

由右线性文法转换得到正规表达式的规则为：

(1)形如$U \to \alpha V$, $V \to \beta$的产生式转换成正规表达式$U=\alpha\beta$;

(2)形如$U \to \alpha U|\beta$的产生式转换成$U=\alpha^*\beta$;

(3)形如$U \to \alpha|\beta$的产生式转换成$U=\alpha|\beta$。

(4)反复使用(1)、(2)、(3)，直到文法只剩下一条关于文法开始符号的产生式，且该条产生式的右部不含非终结符号。这个产生式的右部就是正规表达式。

RG ⇒ RE举例

[例] 求与下列文法G[S]等价的正规表达式。

（1） S::=aS|aB

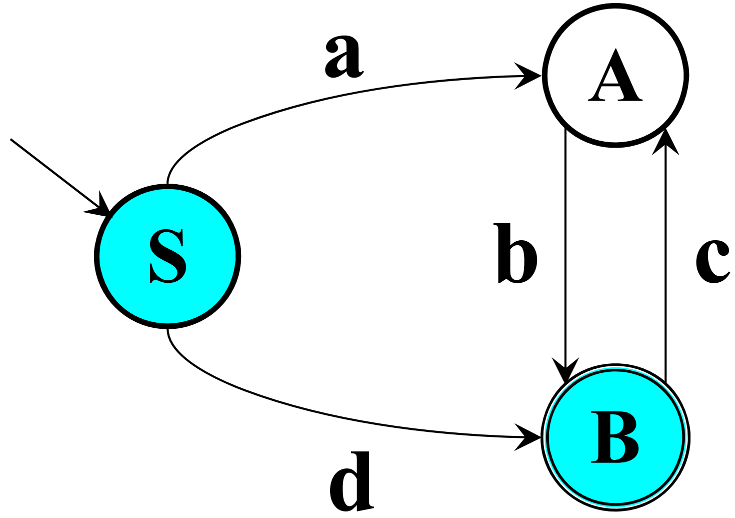
B::=bB|bA

A::=cA|c

（2） S::=dA|eB

A::=aA|b

B::=bB|c



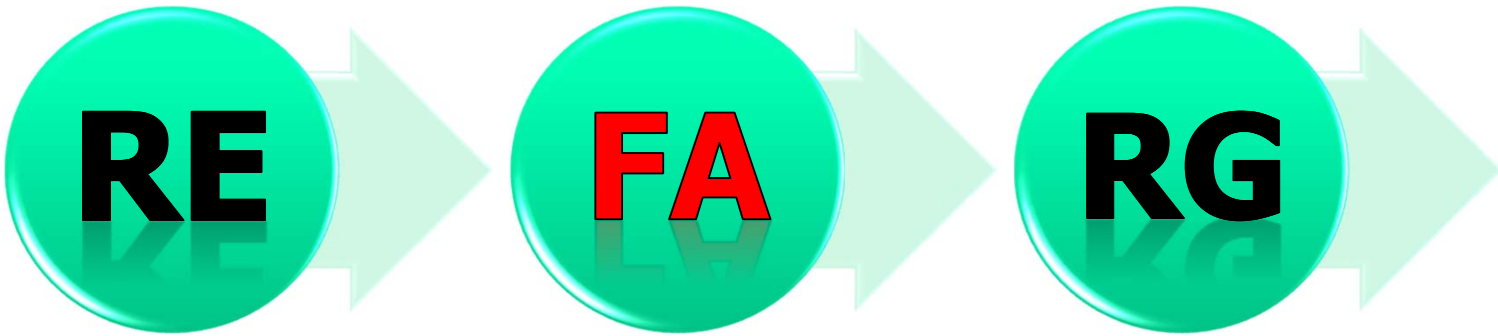
（3） S::=aA

A::=aA|bA|d

$RE \Rightarrow RG$

2. RE→RG

一般地，直接由RE写RG有困难，但可通过FA写出。



RG ⇒ RE

2. RE→RG

(1) 令RG为G[S]，对RE e，形成产生式S→e；

(2) 利用下列替换规则，重写产生式，直至符合RG形式要求：

① $A \to xy$ 替换成 $A \to xB$， $B \to y$ (新增 $B \in V_N$)

② $A \to x^*y$ 替换成 $A \to xA \mid y$

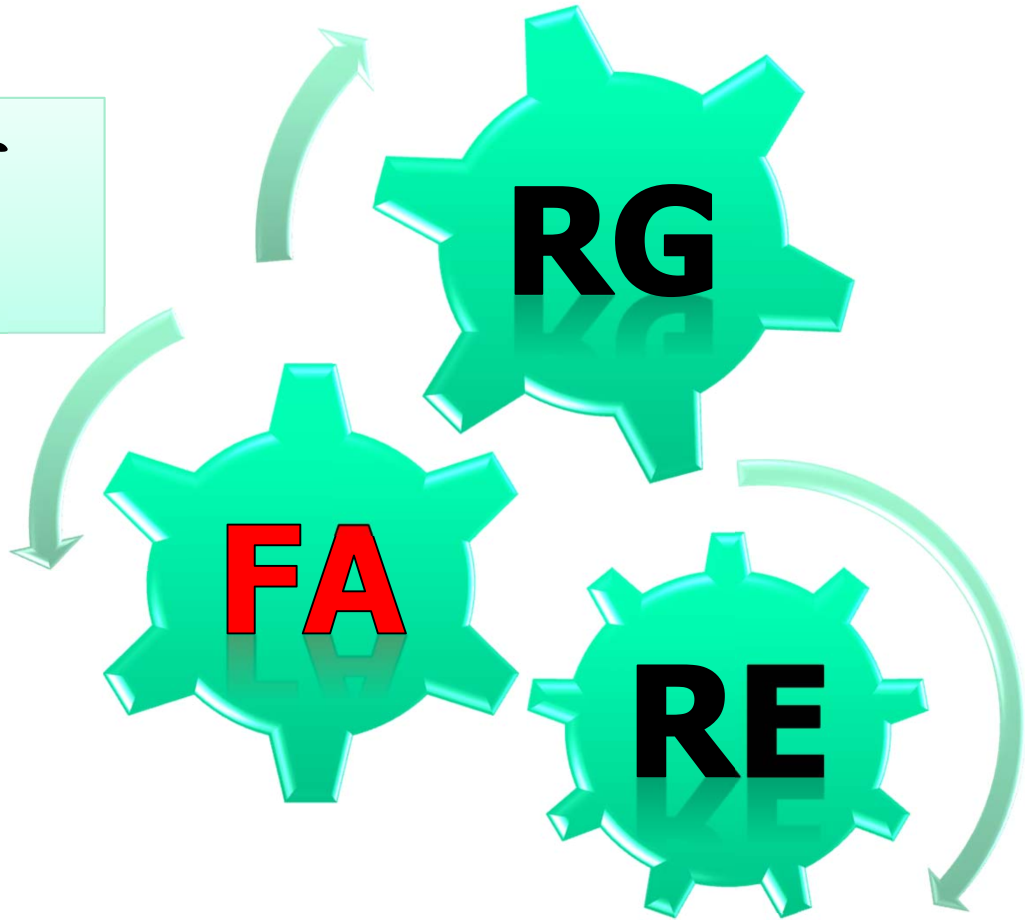
③ $A \to x \mid y$ 替换成 $A \to x$， $A \to y$

即得所求之RG G[S]。

【例】将(a|b)*a转换成等价的RG。

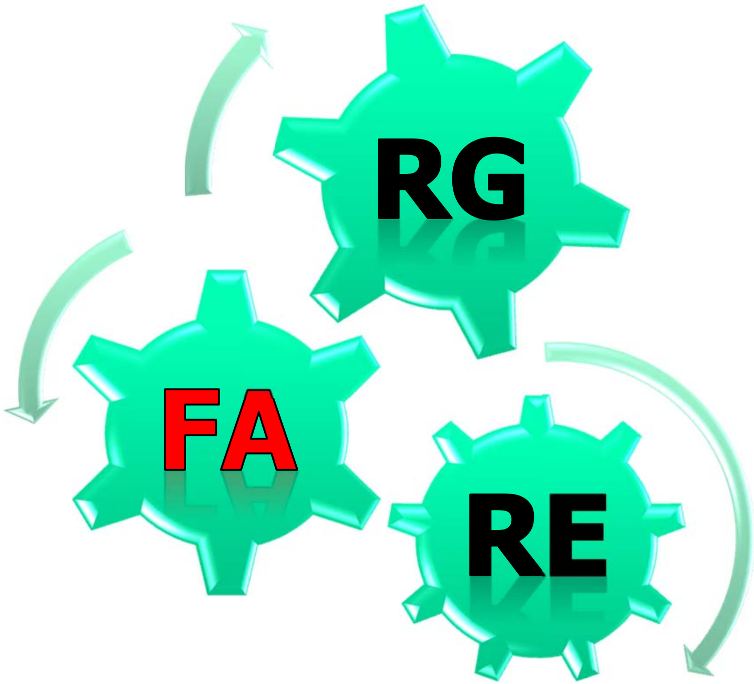
RG、FA和RE的关系

正规语言
（正规集）



RG、FA和RE的关系

正规语言
（正规集）



THEOREM

The following statements are equivalent:

(1) L is a regular set.

(2) L is a right-linear language.

(3) L is a finite automaton language.

(4) L is a nondeterministic finite automaton language.

(5) L is denoted by a regular expression.

第3章内容小结

- DFA的形式定义
- NDFA的形式定义
- FA的构造
- 由NDFA构造DFA
- DFA最小化
- 正规表达式
- 由正规表达式构造FA
- 由FA构造正规表达式

下章内容简介 —— 第4章

- 词法分析程序与单词符号
- 词法分析程序的设计
- 词法分析程序的自动生成