


第三章 词法分析及其自动构造



词法分析程序的设计原则，单词的描述技术，识别机制及词法分析程序的自动构造原理。

- 
- 单词的描述工具
 - 单词的识别系统
 - 设计词法分析程序,实现词法分析程序的自动构造



教学要求

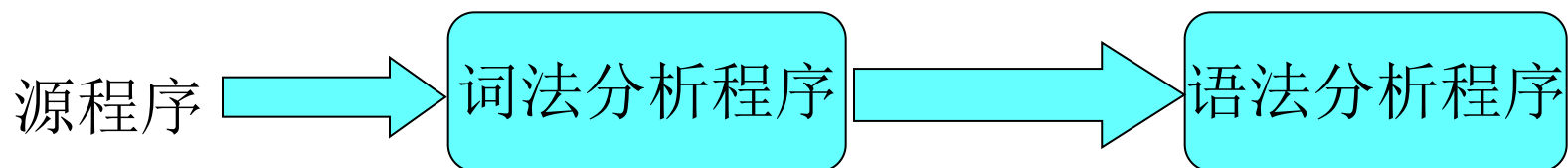
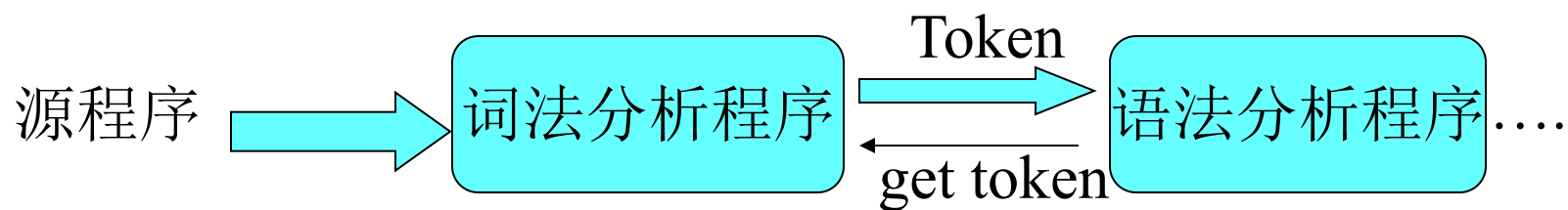
1. 掌握：正规式，DFA的概念，NFA的概念
2. 理解：将DFA 转换为NFA，正规式与有穷自动机间的转换


回顾 什么是词法分析程序

⌘ 实现词法分析（lexical analysis）的程序

- ☑ 逐个读入源程序字符并按照构词规则切分成一系列单词。
- ☑ 单词是语言中具有独立意义的最小单位，包括保留字、标识符、运算符、标点符号和常量等。
- ☑ 词法分析是编译过程中的一个阶段，在语法分析前进行。也可以和语法分析结合在一起作为一遍，由语法分析程序调用词法分析程序来获得当前单词供语法分析使用。

词法分析程序和语法分析程序的关系





词法分析程序的**主要任务**:


☒ 读源程序，产生单词符号

词法分析程序的**其他任务**:

☒ 滤掉空格，跳过注释、换行符


☒ 追踪换行标志，复制出错源程序，

☒ 宏展开，



词法分析工作从语法分析工作独立出来的原因：

- ☑ 简化设计
- ☑ 改进编译效率
- ☑ 增加编译系统的可移植性

- 
- 单词的描述工具-正规表达式, 正规文法
 - 单词的识别系统-有穷自动机
 - 设计词法分析程序,实现词法分析程序的自动构造

正规式




正规式也称正则表达式,是定义正规集的数学工具。正规表达式 (**regular expression**) 是说明单词的模式(**pattern**)的一种重要的表示法 (记号), 我们用以描述单词符号。




定义（正规式和它所表示的正规集）：

设字母表为 Σ ，辅助字母表 $\Sigma' = \{\Phi, \varepsilon, |, \bullet, *, (,)\}$ 。

- 1 ε 和 Φ 都是 Σ 上的正规式，它们所表示的正规集分别为 $\{\varepsilon\}$ 和 $\{\ }$ ；

- 
- 2 任何 $a \in \Sigma$, a 是 Σ 上的一个正规式, 它所表示的正规集为 $\{a\}$;
- 3 假定 e_1 和 e_2 都是 Σ 上的正规式, 它们所表示的正规集分别为 $L(e_1)$ 和 $L(e_2)$, 那么, (e_1) , $e_1 \mid e_2$, $e_1 \bullet e_2$, e_1^* 也都是正规式, 它们所表示的正规集分别为 $L(e_1)$, $L(e_1) \cup L(e_2)$, $L(e_1)L(e_2)$ 和 $(L(e_1))^*$ 。



4 仅由有限次使用上述三步骤而定义的表达式才是 Σ 上的正规式，仅由这些正规式所表示的集合才是 Σ 上的正规集。

正规式中的符号

其中的“|”读为“或”；“●”读为“连接”；“*”读为“闭包”（即，任意有限次的自重复连接）。在不致混淆时，括号可省去，但规定算符的优先顺序为“*”、“●”、“|”。连接符“●”一般可省略不写。“*”、“●”和“|”都是左结合的。

令 $\Sigma = \{a, b\}$, Σ 上的正规式和相应的正规集的例子

a

$\{a\}$

$a \mid b$

$\{a, b\}$

ab

$\{ab\}$

$(a \mid b)(a \mid b)$

$\{aa, ab, ba, bb\}$

a^*

$\{\epsilon, a, aa, \dots \text{任意个 } a \text{ 的串}\}$

$(a \mid b)^*$

$\{\epsilon, a, b, aa, ab, bb, \dots \text{所有由 } a \text{ 和 } b \text{ 组成的串}\}$

$(a \mid b)^*(aa \mid bb)(a \mid b)^*$

$\{\Sigma^* \text{ 上所有含有两个相继的 } a \text{ 或两个相继的 } b \text{ 组成的串}\}$

讨论两个例子

例4.1

令 $\Sigma = \{l, d\}$, 则 Σ 上的正规式 $r = l(l|d)^*$ 定义的正规集为: $\{l, ll, ld, ldd, \dots\}$, 其中 l 代表字母, d 代表数字, 正规式 即是 字母(字母|数字)*, 它表示的正规集中的每个元素的模式是“字母打头的字母数字串”, 就是 **Pascal** 和多数程序设计语言允许的标识符的词法规则.

讨论两个例子

例4.2

$\Sigma = \{d, \bullet, e, +, -\}$,

则 Σ 上的正规式

$$d^*(\bullet dd^* \mid \varepsilon)(e(+ \mid - \mid \varepsilon)dd^* \mid \varepsilon)$$

表示的是无符号数的集合。其中d为0~9的数字。

2, 12.59, 3.6e2, 471.88e-1

程序设计语言的单词都能用正规式来定义。

若两个正规式 e_1 和 e_2 所表示的正规集相同,
则说 e_1 和 e_2 等价, 写作 $e_1=e_2$ 。

☞ 例如: $e_1 = a \mid b$, $e_2 = b \mid a$

☞ 又如: $e_1 = b(ab)^*$, $e_2 = (ba)^*b$
 $e_1 = (a \mid b)^*$, $e_2 = (a^* \mid b^*)^*$

⌘ 设 r, s, t 为正规式，正规式服从的代数规律有：

$$1 \quad r \mid s = s \mid r$$

“或” 服从交换律

$$2 \quad r \mid (s \mid t) = (r \mid s) \mid t$$

“或” 的可结合律

$$3 \quad (rs)t = r(st)$$

“连接” 的可结合律

$$4 \quad r(s \mid t) = rs \mid rt$$

$$(s \mid t)r = sr \mid tr \quad \text{分配律}$$

$$5 \quad \varepsilon r = r, \quad r\varepsilon = r$$

ε 是“连接”的恒等元素

$$6 \quad r \mid r = r$$

$$r^* = \varepsilon \mid r \mid rr \mid \dots$$

“或”的抽取律



程序设计语言中的单词都能用正规式来定义

例子1



⌘ $\Sigma = \{0, 1\}$ 上, 至少有两个连续的0的0和1串的集合?


⌘ 以11结尾的0和1串的集合?

例子1



$$\begin{aligned} & (0|1)^* 00 (0|1)^* \\ & \quad (0|1)^* 11 \end{aligned}$$

例子2

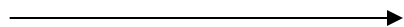


⌘ $\Sigma = \{0, 1\}$ 上，以0开始以1结尾的
的0和1串的集合？

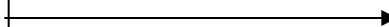
识别



⌘01010



黑盒子



是、否？

有穷自动机

- ⌘ 有穷自动机(也称有限自动机)作为一种识别装置，它能准确地识别正规集，即识别正规式所表示的集合。应用有穷自动机这个理论，为词法分析程序的自动构造寻找有效的方法和工具。
- ⌘ 有穷自动机分为两类：确定的有穷自动机(Deterministic Finite Automata, DFA)和不确定的有穷自动机(Nondeterministic Finite Automata, NFA)。

关于有穷自动机我们将讨论如下题目



确定的有穷自动机DFA

不确定的有穷自动机NFA

NFA的确定化

DFA的最小化

确定的有穷自动机**DFA**

DFA定义：

一个确定的有穷自动机（**DFA**）**M**是一个五元组：

M = (**K**, Σ , **f**, **S**, **Z**) 其中

1. **K**是一个有穷集，它的每个元素称为一个状态；
2. Σ 是一个有穷字母表，它的每个元素称为一个输入符号，所以也称 Σ 为输入符号表；

DFA定义

3. f 是转换函数，是在 $K \times \Sigma \rightarrow K$ 上的映射，即，如 $f(k_i, a) = k_j$ ，($k_i \in K, k_j \in K$)就意味着，当前状态为 k_i ，输入符为 a 时，将转换为下一个状态 k_j ，我们把 k_j 称作 k_i 的一个后继状态；
4. $S \in K$ 是唯一的一个初态；
5. $Z \subset K$ 是一个终态集，终态也称可接受状态或结束状态。

一个**DFA** 的例子:

DFA $M = (\{S, U, V, Q\}, \{a, b\}, f, S, \{Q\})$ 其中 f 定义为:

$$f(S, a) = U$$

$$f(V, a) = U$$

$$f(S, b) = V$$

$$f(V, b) = Q$$

$$f(U, a) = Q$$

$$f(Q, a) = Q$$

$$f(U, b) = V$$

$$f(Q, b) = Q$$

DFA 的状态图表示

$$f(S, a) = U$$

$$f(S, b) = V$$

$$f(U, a) = Q$$

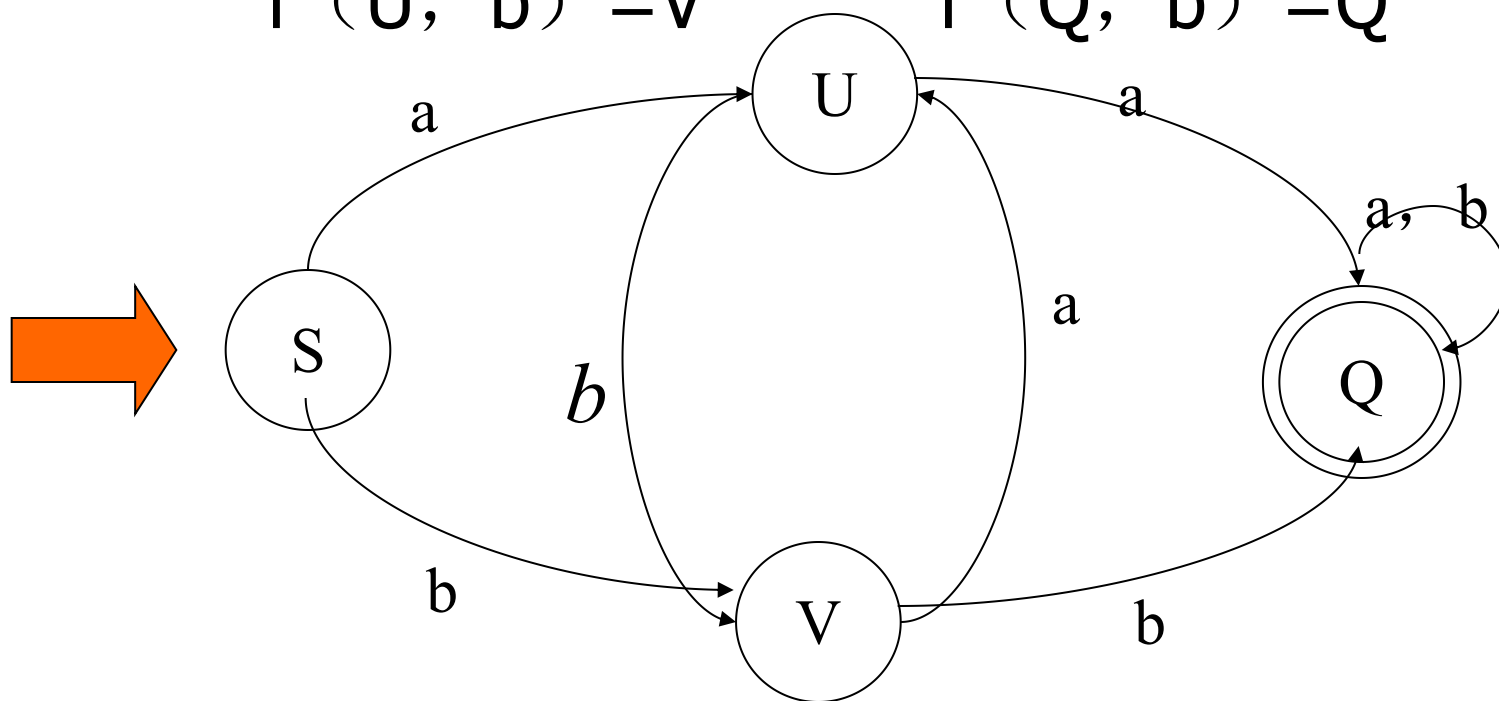
$$f(U, b) = V$$

$$f(V, a) = U$$

$$f(V, b) = Q$$

$$f(Q, a) = Q$$

$$f(Q, b) = Q$$



DFA 的矩阵表示

状态 \ 字符	a	b
S	U	V
U	Q	V
V	U	Q
Q	Q	Q

0
0
0
1

为了说明DFA如何作为一种识别机制,
我们还要理解下面的定义

Σ^* 上的符号串 t 在DFA M 上运行

一个输入符号串 t , (将它表示成 Tt_1 的形式,
其中 $T \in \Sigma$, $t_1 \in \Sigma^*$) 在DFA $M = (K, \Sigma, f, S, Z)$ 上运行的定义为:

$f(Q, Tt_1) = f(f(Q, T), t_1)$ 其中 $Q \in K$

扩充转换函数 f 为 $K \times \Sigma^* \rightarrow K$ 上的映射, 且:

$$f(k_i, \varepsilon) = k_i$$

Σ^* 上的符号串 t 被DFA M 接受

$$M = (K, \Sigma, f, S, Z)$$

若 $t \in \Sigma^*$, $f(S, t) = P$, 其中 S 为
 M 的开始状态, $P \in Z$, Z 为终态
集。

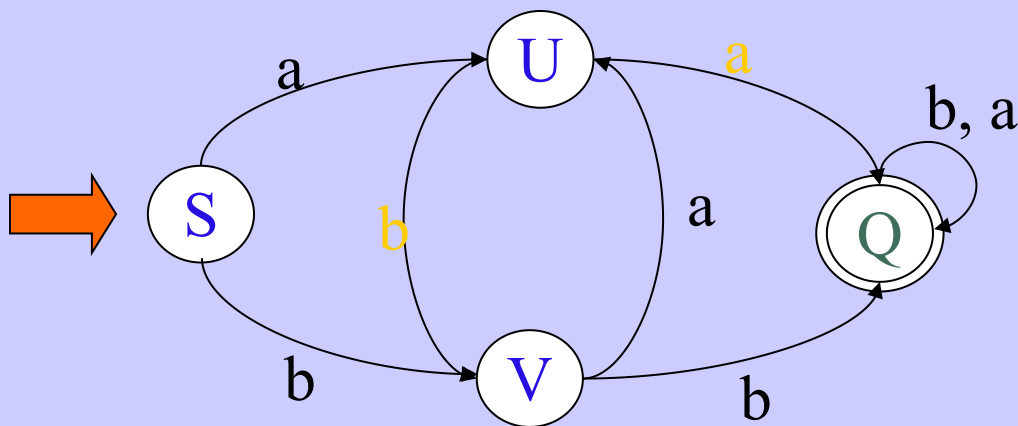
则称 t 为DFA M 所接受（识别）。

Σ^* 上的符号串t被DFA M接受

例：证明**t=baab**被下图的**DFA**所接受。

$$\begin{aligned} f(S, baab) &= f(f(S, b), aab) \\ &= f(V, aab) = f(f(V, a), ab) \\ &= f(U, ab) = f(f(U, a), b) \\ &= f(Q, b) = Q \end{aligned}$$

Q属于终态。
得证。



DFA M 所能接受的符号串的全体记为 $L(M)$.

结论:

Σ 上一个符号串集 $V \subset \Sigma^*$ 是正规的, 当且仅当
存在一个 Σ 上的确定有穷自动机 M , 使得

$$V = L(M).$$

DFA的确定性表现在转换函数

$f: K \times \Sigma \rightarrow K$ 是一个单值函数，也就是说，对任何状态 $k \in K$ ，和输入符号 $a \in \Sigma$ ， $f(k, a)$ 唯一地确定了下一个状态。从状态转换图来看，若字母表 Σ 含有 n 个输入字符，那末任何一个状态结点最多有 n 条弧射出，而且每条弧以一个不同的输入字符标记。

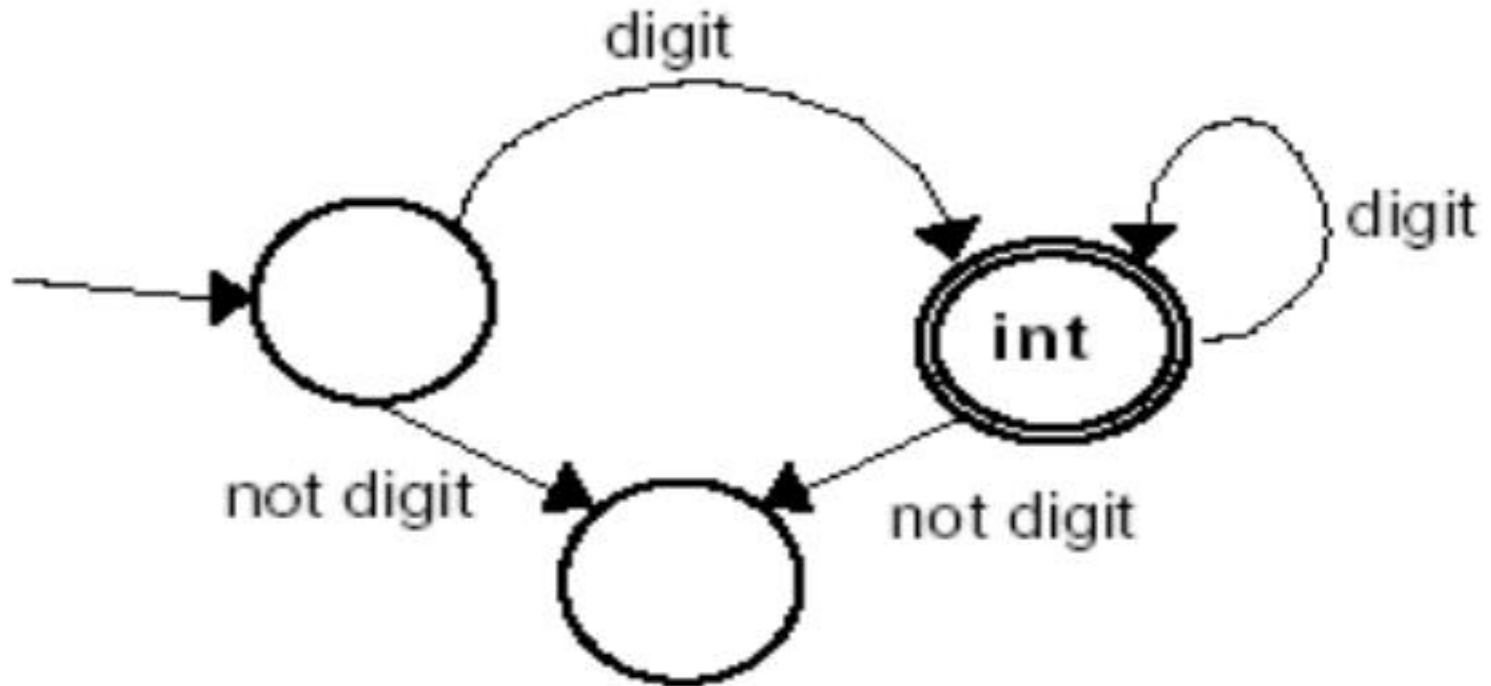
DFA的行为很容易用程序来模拟.

DFA $M = (K, \Sigma, f, S, Z)$ 的行为的模拟程序

```
⊗ K:=S;  
⊗ c:=getchar;  
⊗ while c<>eof do  
⊗   {K:=f(K,c);  
⊗     c:=getchar;  
⊗   };  
⊗ if K is in Z then return ('yes')  
⊗           else return ('no')
```

DFA

$\Sigma = \{\underline{\text{digit}}, \underline{\text{not digit}}\}$



不确定的有穷自动机**NFA**

定义

NFA $M = \{K, \Sigma, f, S, Z\}$, 其中 K 为状态的有穷非空集, Σ 为有穷输入字母表, f 为 $K \times \Sigma^*$ 到 K 的子集 (2^K) 的一种映射, $S \subseteq K$ 是初始状态集, $Z \subseteq K$ 为终止状态集.

例子

NFA $M = (\{S, P, Z\}, \{0, 1\}, f, \{S, P\}, \{Z\})$

其中

$f(S, 0) = \{P\}$

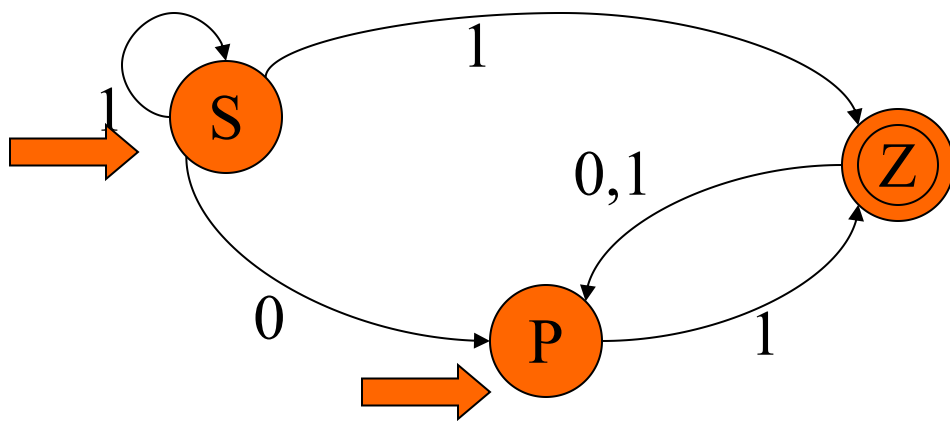
$f(S, 1) = \{S, Z\}$

$f(P, 1) = \{Z\}$

$f(Z, 0) = \{P\}$

$f(Z, 1) = \{P\}$

状态图表示



矩阵表示

矩阵表示

	0	1	
S	{P}	{S,Z}	0
P	{}	{Z}	0
Z	{P}	{P}	1

简化为



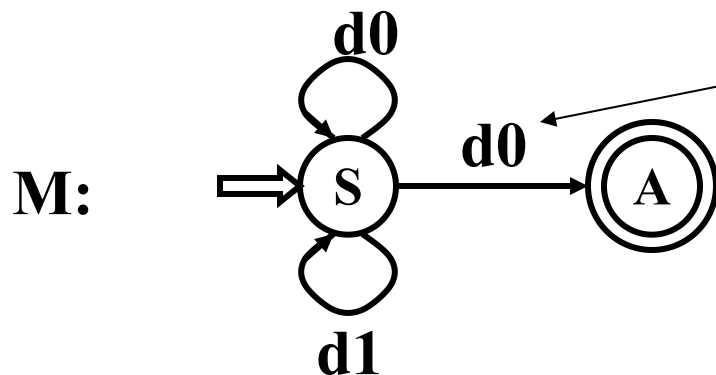
	0	1	
S	P	S,Z	0
P	.	Z	0
Z	P	P	1

NFA的例子-1

很多情况下，当从语言构造一个自动机时，一般会先构造一个NFA。

例1：构造自动机M，使其接受的语言是偶整数串的集合

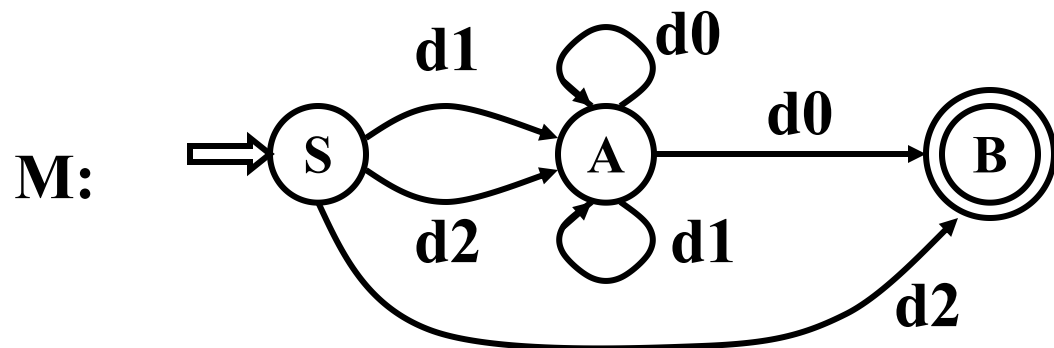
设 $d0 = \{ 0, 2, 4, 6, 8 \}$ $d1 = \{ 1, 3, 5, 7, 9 \}$



一般，每条边上只允许有一个符号。这里是一种简化的表示。

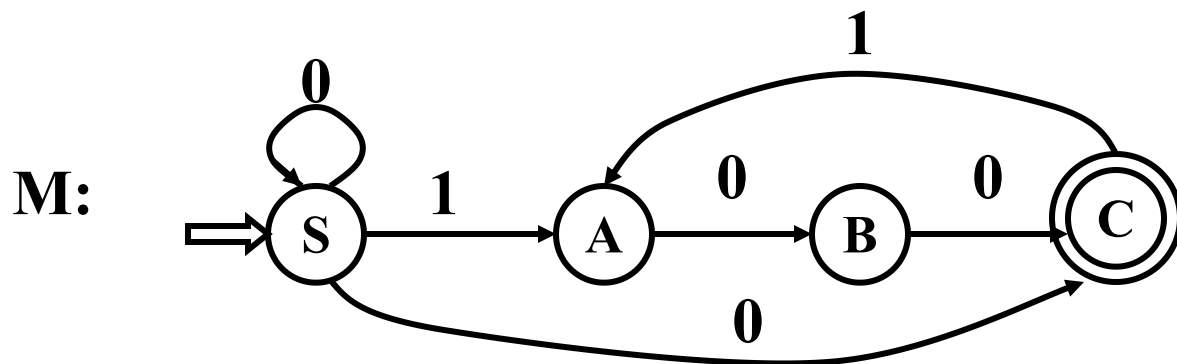
例2：构造自动机M，使其接受的语言是偶整数串的集合，但最高位不允许为0

设 $d0 = \{ 0, 2, 4, 6, 8 \}$ $d1 = \{ 1, 3, 5, 7, 9 \}$ $d2 = \{ 2, 4, 6, 8 \}$



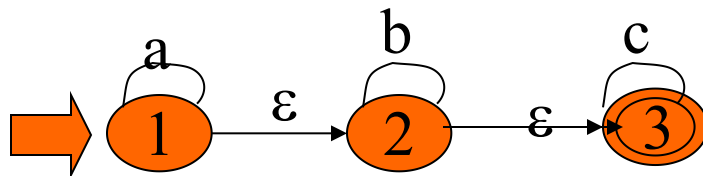
NFA的例子-2

例：设有语言 $L = \{ w \mid w \in \{0, 1\}^+ \text{, 并且 } w \text{ 中的每个 } 1 \text{ 后面恰有两个相继的 } 0 \text{ 直接跟随} \}$ ，构造接受这个语言的自动机 M 。



具有 ϵ 转移的不确定的有穷自动机

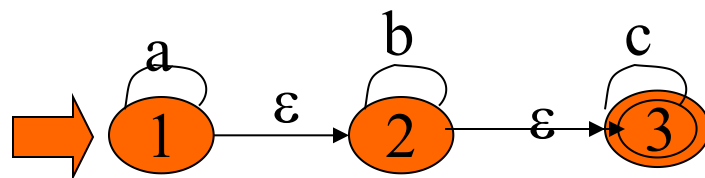
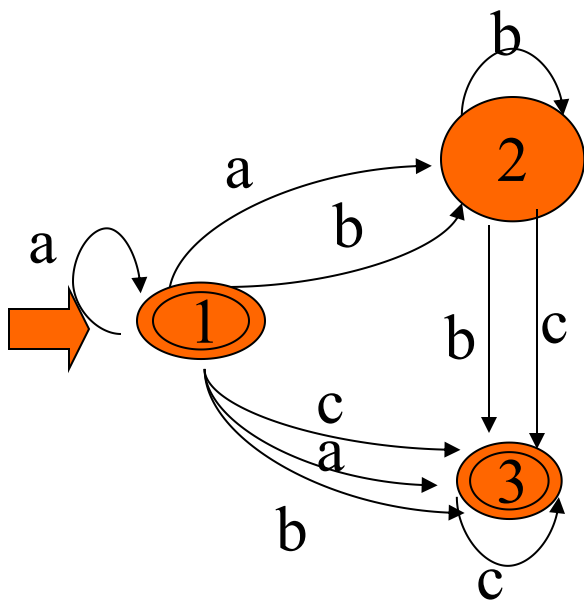
f 为 $K \times \Sigma^*$ 到 K 的子集 (2^K) 的一种映射



有如下定理：

对任何一个具有 ϵ 转移的不确定的有穷自动机
NFA N ，一定存在一个不具有 ϵ 转移的不确
定的有穷自动机NFA M ，使得 $L(M)=L(N)$ 。

与上例等价的一个NFA.



类似**DFA**, 对**NFA** $M = \{K, \Sigma, f, S, Z\}$ 也有如下定义

Σ^* 上的符号串 t 在NFA M 上运行..

一个输入符号串 t , (我们将它表示成 Tt_1 的形式, 其中 $T \in \Sigma, t_1 \in \Sigma^*$) 在NFA M 上运行的定义为:

$f(Q, Tt_1) = f(f(Q, T), t_1)$ 其中 $Q \in K$.

Σ^* 上的符号串 t 被NFA M 接受

若 $t \in \Sigma^*$, $f(S_0, t) = P$, 其中 $S_0 \in S, P \in Z$,

则称 t 为NFA M 所接受 (识别)

Σ^* 上的符号串 t 被NFA M 接受也可以这样理解

对于 Σ^* 中的任何一个串 t ，若存在一条从某一初态到某一终态的道路，且这条道路上所有弧的标记字依序连接成的串(不理睬那些标记为 ε 的弧)等于 t ，则称 t 可为NFA M 所识别(读出或接受)。若 M 的某些结既是初态又是终态，或者存在一条从某个初态到某个终态的道路，其上所有弧的标记均为 ε ，那么空字可为 M 所接受。

000

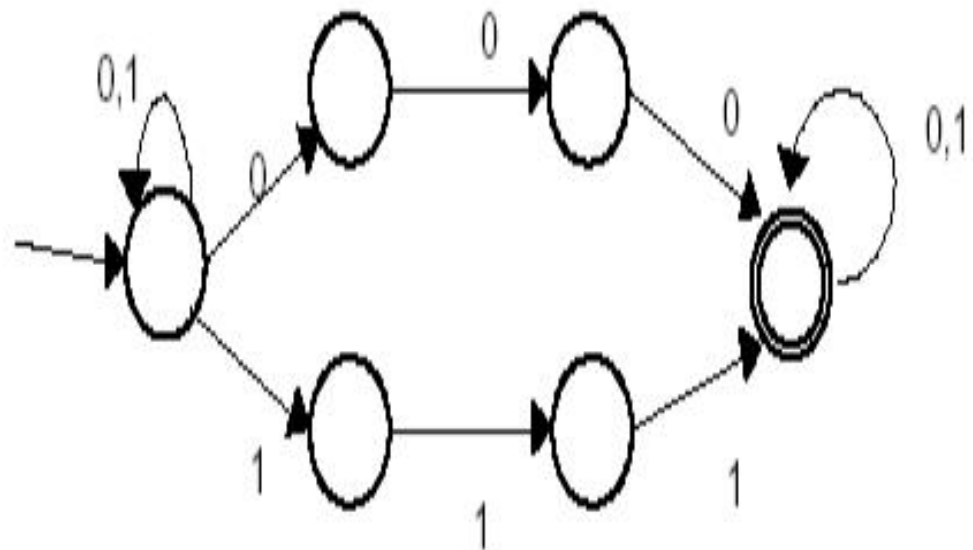
111

1010001

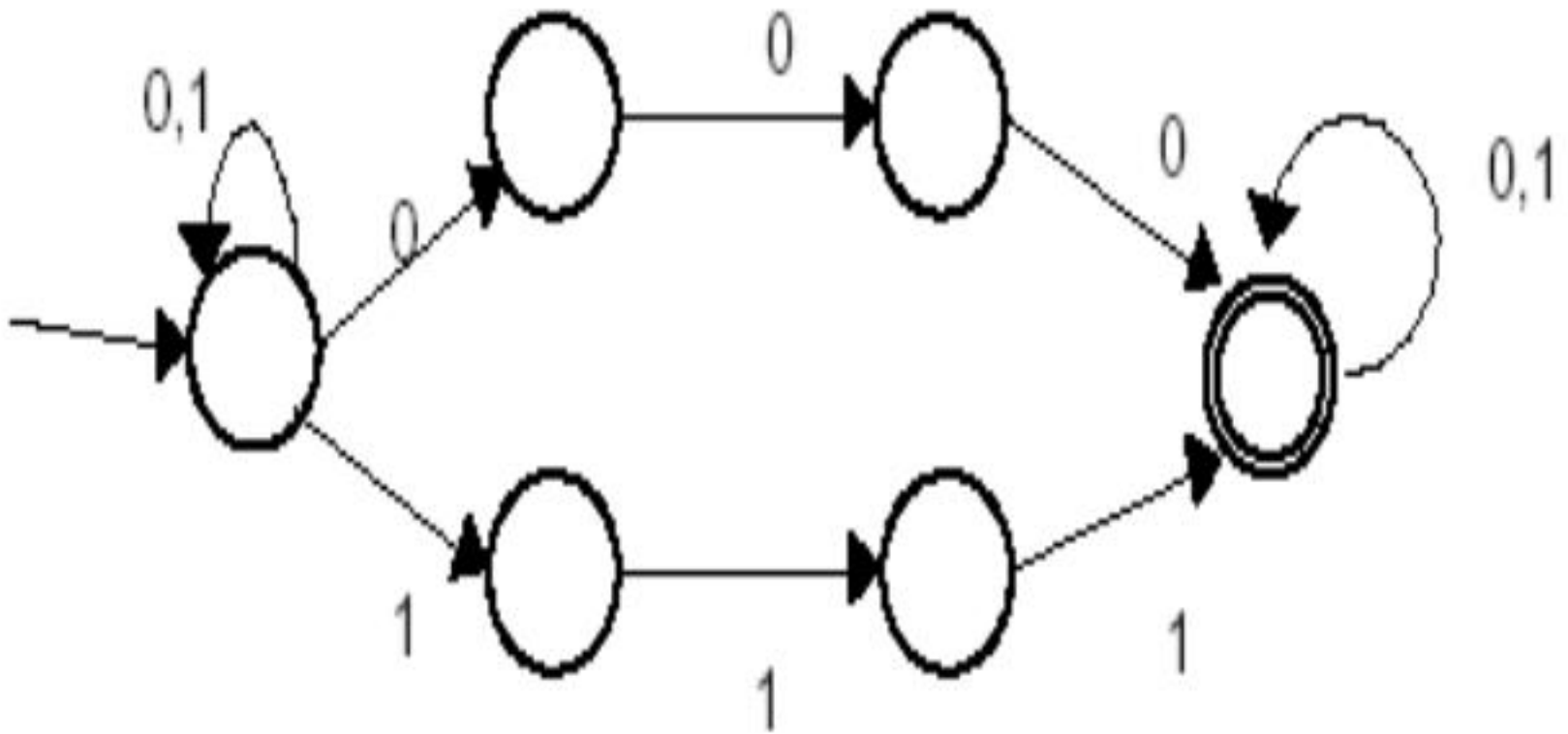
110000001


00

01100



$(0|1)^*(000|111)(0|1)^*$





NFA M 所能接受的符号串的全体记为
 $L(M)$

结论:

Σ 上一个符号串集 $V \subset \Sigma^*$ 是正规的, 存在一个 Σ 上的不确定的有穷自动机 M , 使得 $V = L(M)$ 。

确定有限自动机和不确定有限自动机

无论是NFA还是DFA，它们都是定义正规集合的工具。他们定义集合的能力相同。所以任意一个NFA都可等价地变换为DFA。例如：



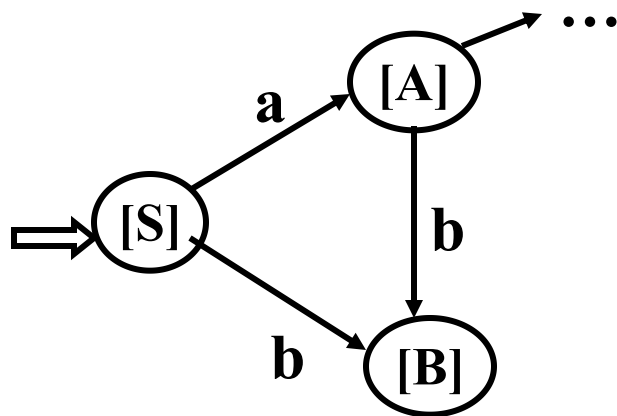
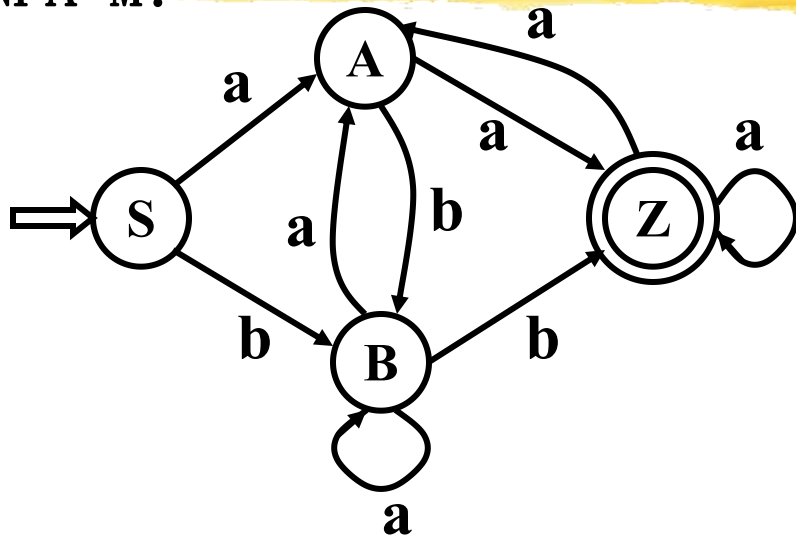
确定有限自动机和不确定有限自动机

DFA是NFA的特例.对每个NFA N 一定存在一个DFA M , 使得 $L(M)=L(N)$ 。对每个NFA N 存在着与之等价的DFA M 。

NFA变换为DFA的例子

无论是NFA还是DFA，它们都是定义正规集合的工具。他们定义集合的能力相同。所以任意一个NFA都可等价地变换为DFA。例如：

NFA M:



⇒

+

+

+

+

状态	a	b
[S]	[A]	[B]
[A]	[Z]	[B]
[B]	[AB]	[Z]
[Z]	[AZ]	
[AB]	[ABZ]	[BZ]
[AZ]	[AZ]	[B]
[ABZ]	[ABZ]	[BZ]
[BZ]	[ABZ]	[Z]

NFA确定化算法:



有一种算法，将NFA转换成接受同样语言的DFA.

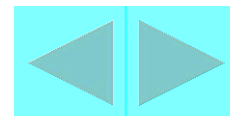
这种算法称为子集法.

与某一NFA等价的DFA不唯一.

NFA确定化算法:

假设NFA $N=(K, \Sigma, f, K_0, K_t)$ 按如下办法构造一个DFA $M=(S, \Sigma, d, S_0, S_t)$, 使得 $L(M)=L(N)$:

1. M 的状态集 S 由 K 的一些子集组成。用 $[S_1 S_2 \dots S_j]$ 表示 S 的元素, 其中 S_1, S_2, \dots, S_j 是 K 的状态。并且约定, 状态 S_1, S_2, \dots, S_j 是按某种规则排列的, 即对于子集 $\{S_1, S_2\}=\{S_2, S_1\}$ 来说, S 的状态就是 $[S_1 S_2]$;



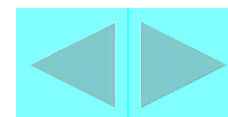
2 M和N的输入字母表是相同的，即是 Σ ；

3 转换函数是这样定义的：

$$d([S_1 S_2 \dots S_j], a) = [R_1 R_2 \dots R_t] \quad \text{其中} \\ \{R_1, R_2, \dots, R_t\} = \varepsilon\text{-closure}(\text{move}(\{S_1, S_2, \dots, S_j\}, a))$$

4 $S_0 = \varepsilon\text{-closure}(K_0)$ 为M的开始状态；

5 $S_t = \{[S_i S_k \dots S_e], \text{ 其中 } [S_i S_k \dots S_e] \in S \text{ 且 } \{S_i, S_k, \dots, S_e\} \cap K_t \neq \Phi\}$



定义对状态集合I的几个有关运算：

1. **状态集合I的 ε -闭包**，表示为 $\varepsilon\text{-closure}(I)$ ，定义为一状态集，是状态集I中的任何状态S经任意条 ε 弧而能到达的状态的集合。

状态集合I的任何状态S都属于 $\varepsilon\text{-closure}(I)$ 。

2. **状态集合I的a弧转换**，表示为 $\text{move}(I,a)$ 定义为状态集合J，其中J是所有那些可从I中的某一状态经过一条a弧而到达的状态的全体。

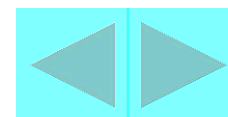
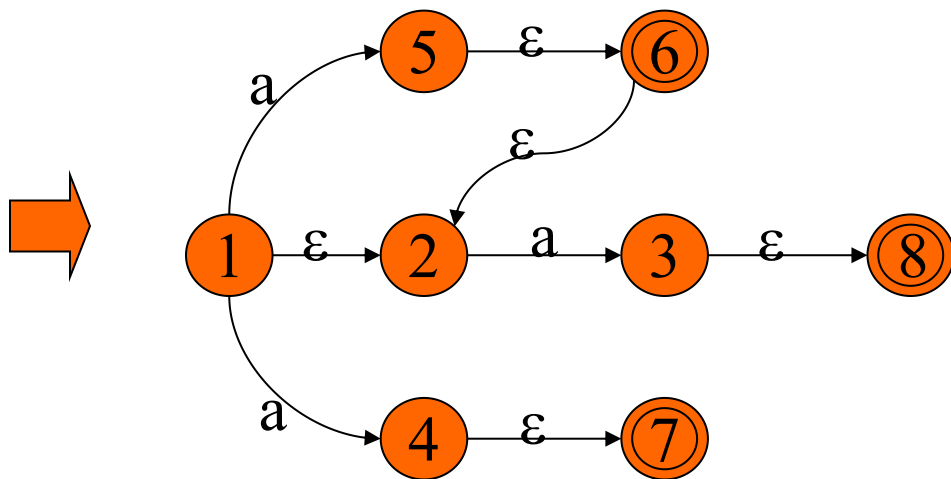
状态集合I的有关运算的例子

$I=\{1\}$, ε -closure(I)= $\{1,2\}$;

$I=\{5\}$, ε -closure(I)= $\{5,6,2\}$;

$\text{move}(\{1,2\},a)=\{5,3,4\}$

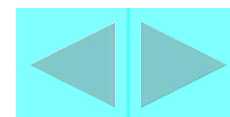
ε -closure($\{5,3,4\}$)= $\{2,3,4,5,6,7,8\}$;



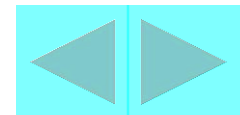
构造NFA N的状态K的子集的算法:

假定所构造的子集族为C, 即 $C = (T_1, T_2, \dots, T_I)$,
其中 T_1, T_2, \dots, T_I 为状态K的子集。

- 1 开始, 令 $\varepsilon\text{-closure}(K_0)$ 为C中唯一成员, 并且它是未被标记的。



2 while (C中存在尚未被标记的子集T) do
 {
 标记T;
 for 每个输入字母a do
 {
 $U := \varepsilon\text{-closure}(\text{move}(T, a));$
 if U不在C中 then
 将U作为未标记的子集加在C中
 }
 }
 }



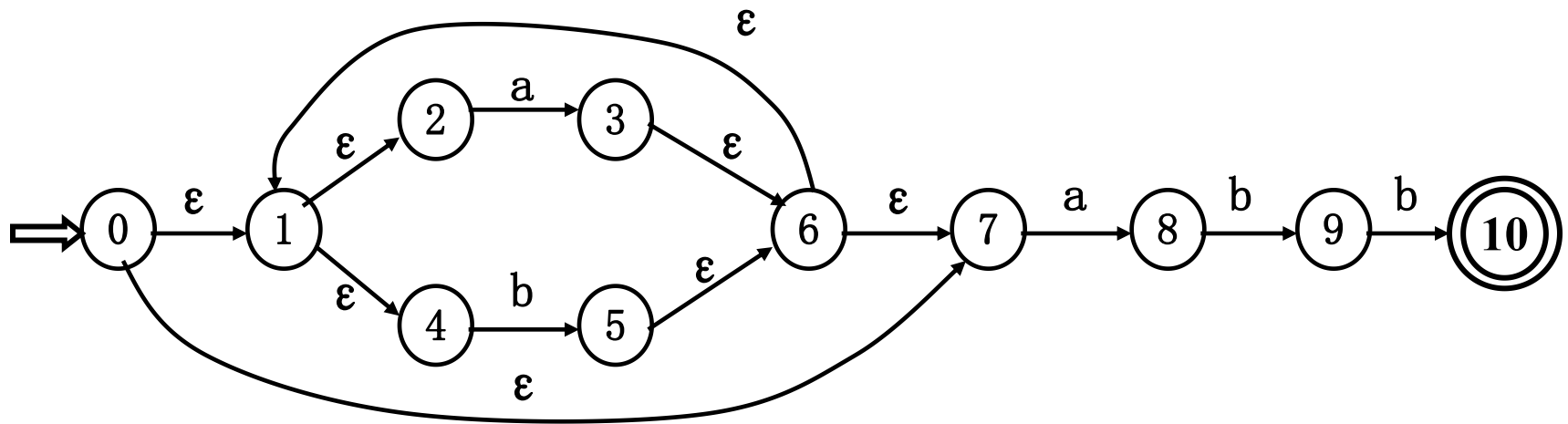
对带空边的NFA的变换—子集构造法

1、状态集合I的 ϵ 闭包 ϵ -closure(I)

ϵ -closure(I): 定义为一个状态集, 是状态I中的任何状态S经任意条 ϵ 弧而能到达的状态集合。

2、状态集合I的a弧转换move(I, a)

move(I, a): 定义为一个状态集合J, 其中J是所有那些可从I中的某一个状态经过一条a弧而能到达的状态的全体。

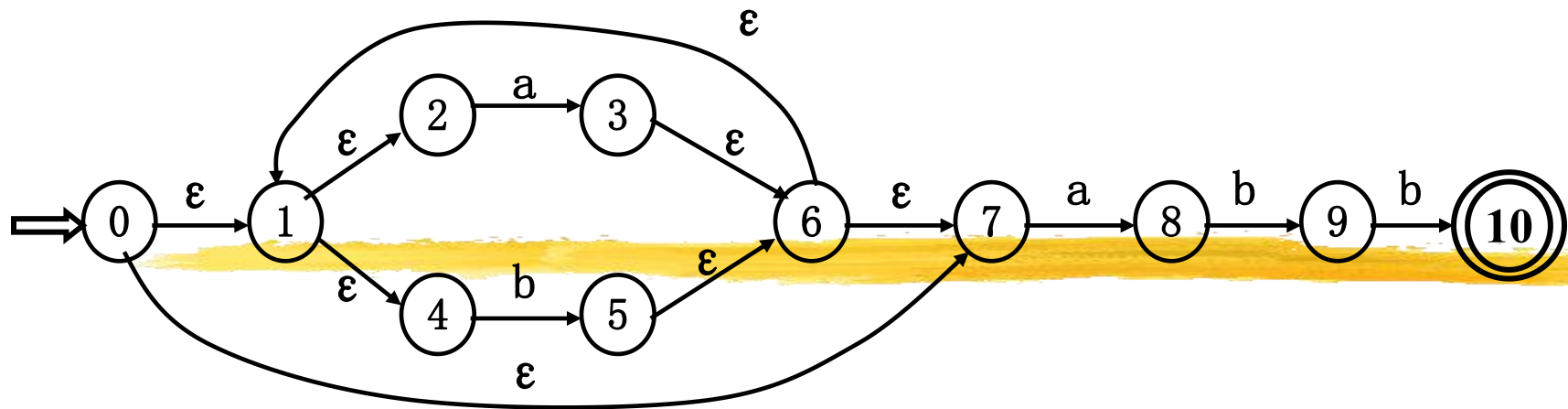


ϵ -closure(0) = { 0, 1, 2, 4, 7 }, 令 $T_0 = \epsilon$ -closure(0)

move(T_0 , a) = { 3, 8 }

由3经 ϵ 边可到达6、7、1、2、4

ϵ -closure(move(T_0 , a)) = ϵ -closure({3, 8})
= { 3, 8, 6, 7, 1, 2, 4 }



$$T_0 = \varepsilon\text{-closure}(0) = \{ 0, 1, 2, 4, 7 \},$$

$$\text{move}(T_0, a) = \{ 3, 8 \}, \quad \text{move}(T_0, b) = \{ 5 \}$$

$$T_1 = \varepsilon\text{-closure}(\text{move}(T_0, a)) = \{ 3, 8, 6, 7, 1, 2, 4 \}$$

$$T_2 = \varepsilon\text{-closure}(\text{move}(T_0, b)) = \{ 5, 6, 7, 1, 2, 4 \}$$

$$\text{move}(T_1, a) = \{ 3, 8 \}, \quad \text{move}(T_1, b) = \{ 5, 9 \}$$

$$\varepsilon\text{-closure}(\text{move}(T_1, a)) = \{ 3, 8, 6, 7, 1, 2, 4 \} = T_1$$

$$\varepsilon\text{-closure}(\text{move}(T_1, b)) = \{ 5, 9, 6, 7, 1, 2, 4 \} = T_3$$

$$\text{move}(T_2, a) = \{ 3, 8 \}, \quad \text{move}(T_2, b) = \{ 5 \}$$

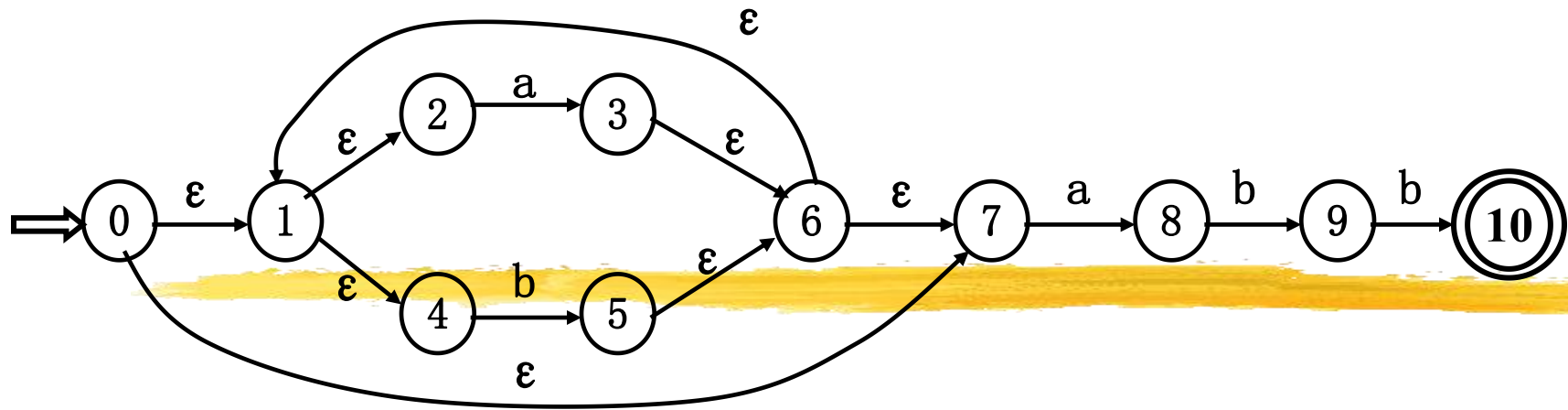
$$\varepsilon\text{-closure}(\text{move}(T_2, a)) = \{ 3, 8, 6, 7, 1, 2, 4 \} = T_1$$

$$\varepsilon\text{-closure}(\text{move}(T_2, b)) = \{ 5, 6, 7, 1, 2, 4 \} = T_2$$

$$\text{move}(T_3, a) = \{ 3, 8 \}, \quad \text{move}(T_3, b) = \{ 5, 10 \}$$

$$\varepsilon\text{-closure}(\text{move}(T_3, a)) = \{ 3, 8, 6, 7, 1, 2, 4 \} = T_1$$

$$\varepsilon\text{-closure}(\text{move}(T_3, b)) = \{ 5, 10, 6, 7, 1, 2, 4 \} = T_4$$



$$\epsilon\text{-closure}(\text{move}(T_3, b)) = \{ 5, 6, 7, 1, 2, 4, 10 \} = T_4$$

$$\text{move}(T_4, a) = \{ 3, 8 \}, \quad \text{move}(T_4, b) = \{ 5 \}$$

$$\epsilon\text{-closure}(\text{move}(T_4, a)) = \{ 3, 8, 6, 7, 1, 2, 4 \} = T_1$$

$$\epsilon\text{-closure}(\text{move}(T_4, b)) = \{ 5, 6, 7, 1, 2, 4 \} = T_2$$

所以，共有如下的状态：

$$T_0 = \{ 0, 1, 2, 4, 7 \}$$

$$T_1 = \{ 1, 2, 3, 4, 6, 7, 8 \}$$

$$T_2 = \{ 1, 2, 4, 5, 6, 7 \}$$

$$T_3 = \{ 1, 2, 4, 5, 6, 7, 9 \}$$

$$T_4 = \{ 1, 2, 4, 5, 6, 7, 10 \}$$

设DFA 的 $V_N = \{T_0, T_1, T_2, T_3, T_4\}$

$$\Sigma = \{a, b\}$$

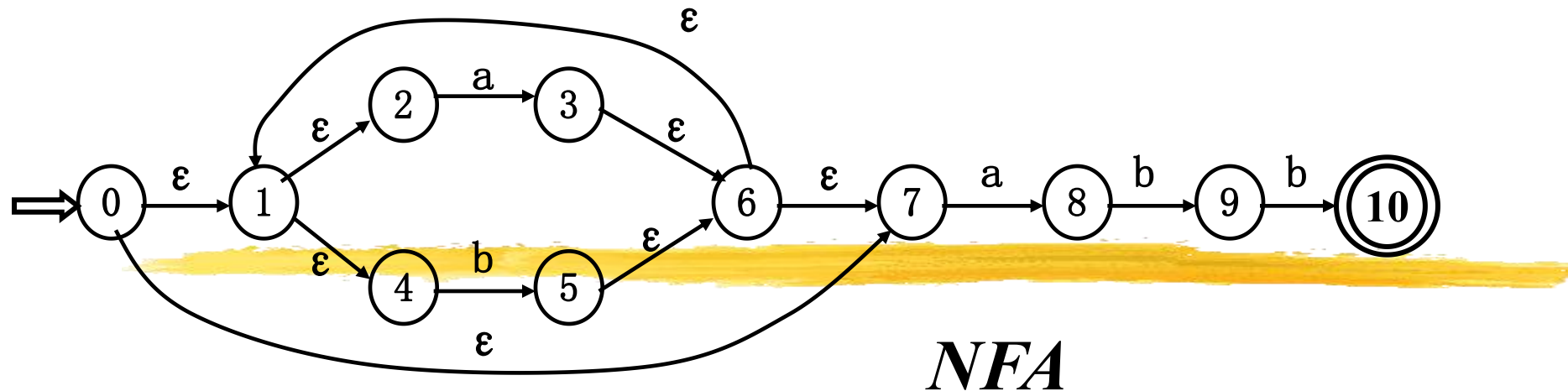
$$D(T_0, a) = T_1, \quad D(T_0, b) = T_2$$

$$D(T_1, a) = T_1, \quad D(T_1, b) = T_3$$

$$D(T_2, a) = T_1, \quad D(T_2, b) = T_2$$

$$D(T_3, a) = T_1, \quad D(T_3, b) = T_4$$

$$D(T_4, a) = T_1, \quad D(T_4, b) = T_2$$



设DFA 的 $V_N = \{T_0, T_1, T_2, T_3, T_4\}$

$\Sigma = \{a, b\}$

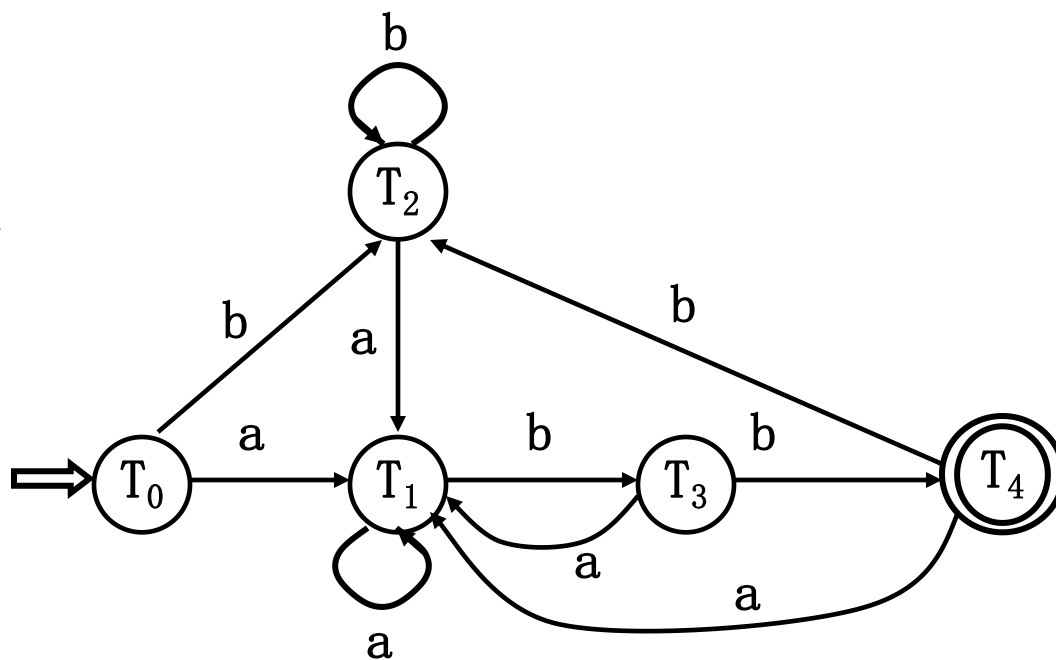
$D(T_0, a) = T_1, \quad D(T_0, b) = T_2$

$D(T_1, a) = T_1, \quad D(T_1, b) = T_3$

$D(T_2, a) = T_1, \quad D(T_2, b) = T_2$

$D(T_3, a) = T_1, \quad D(T_3, b) = T_4$

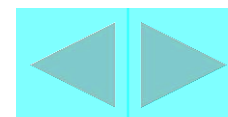
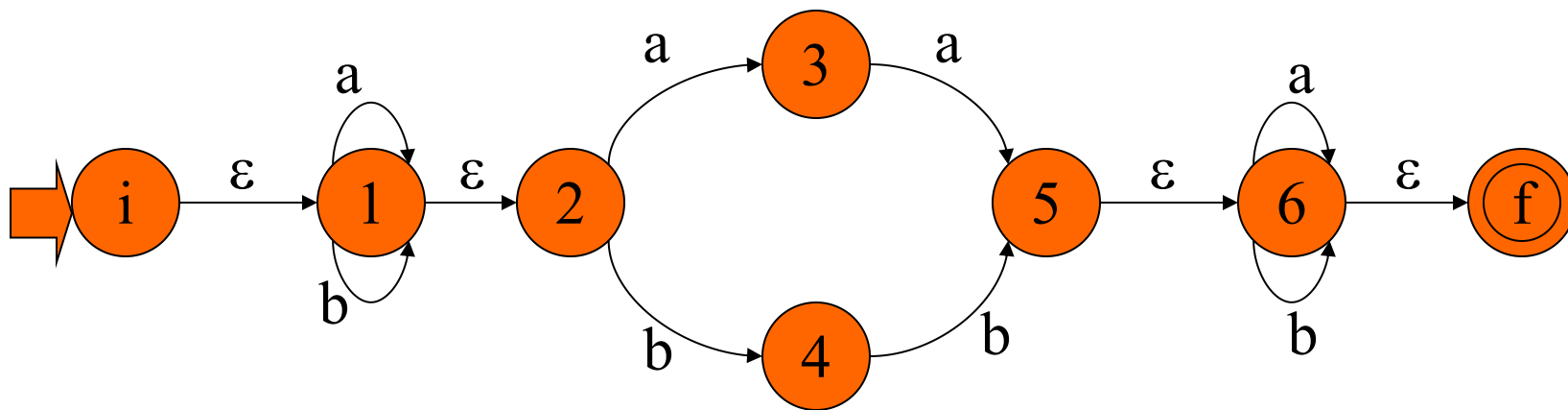
$D(T_4, a) = T_1, \quad D(T_4, b) = T_2$

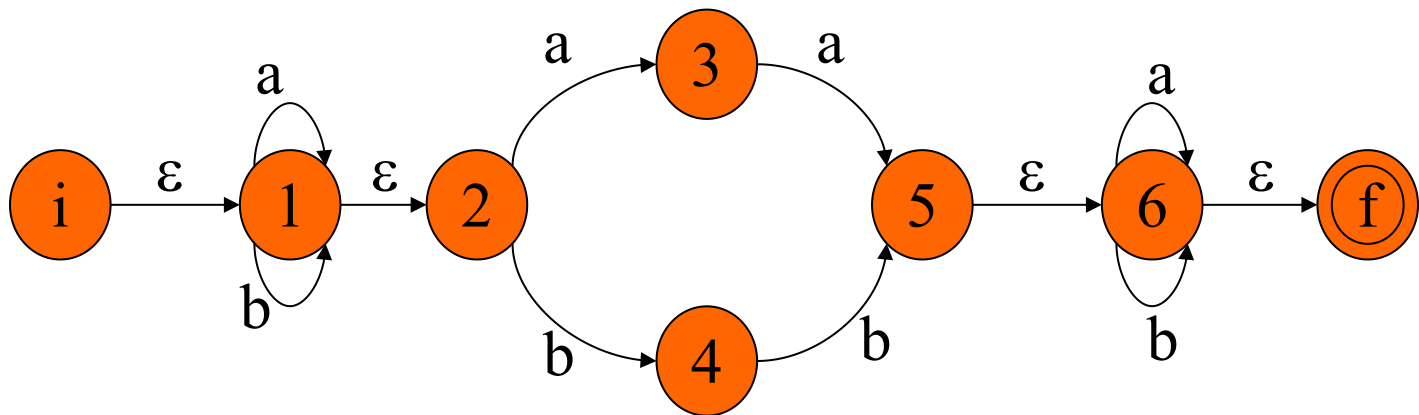


DFA

NFA的确定化

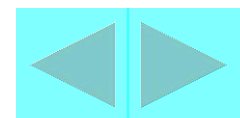
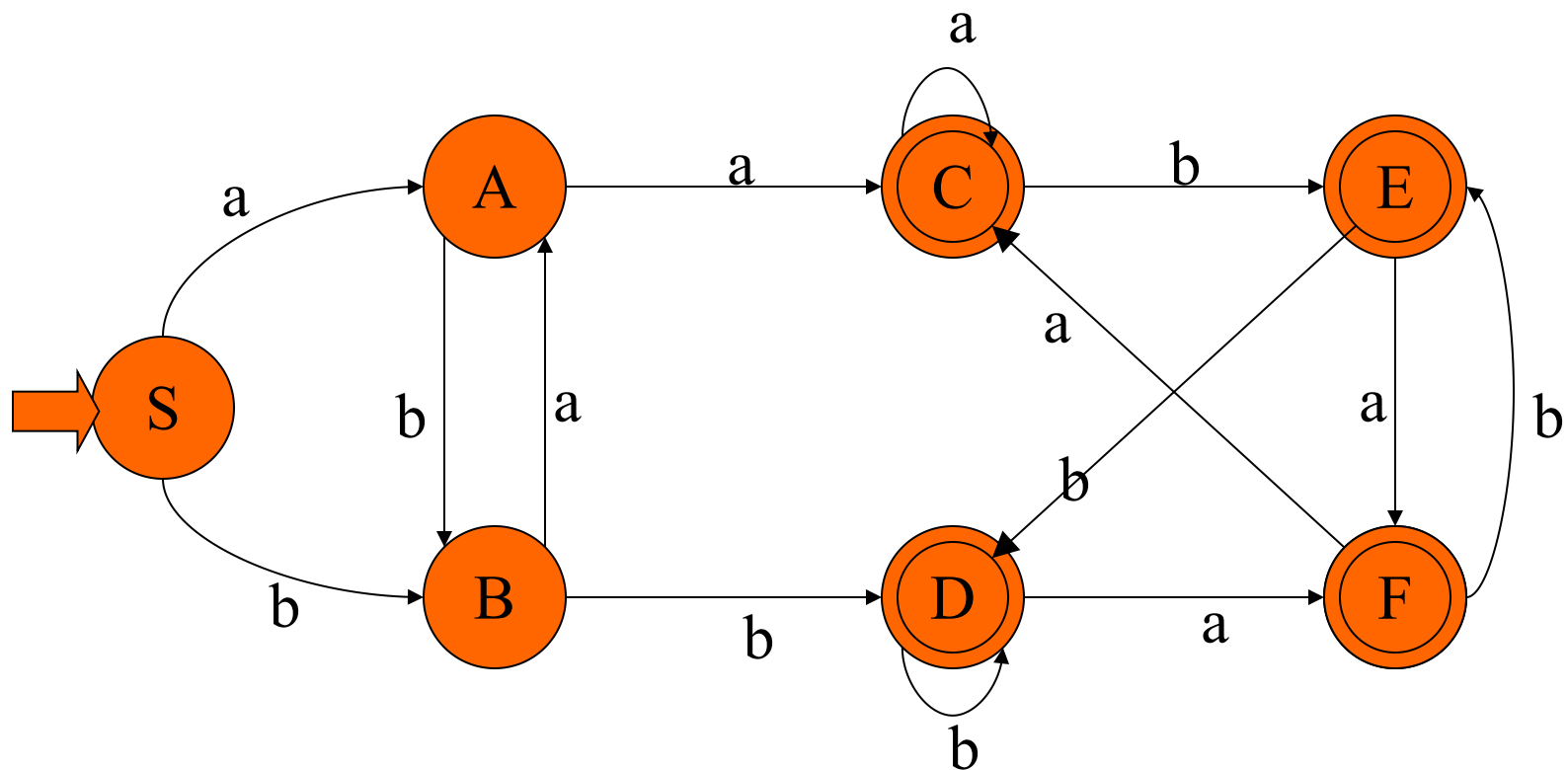
例子





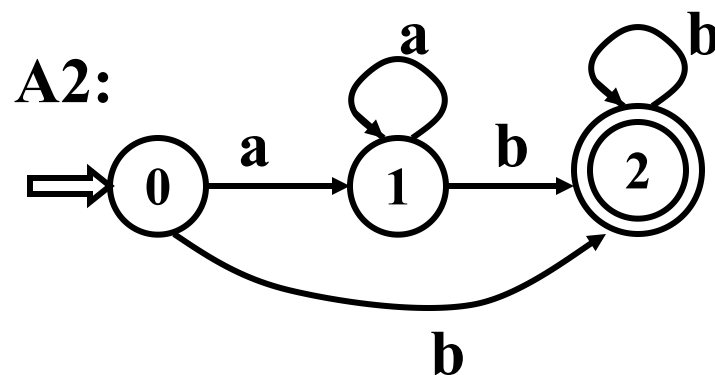
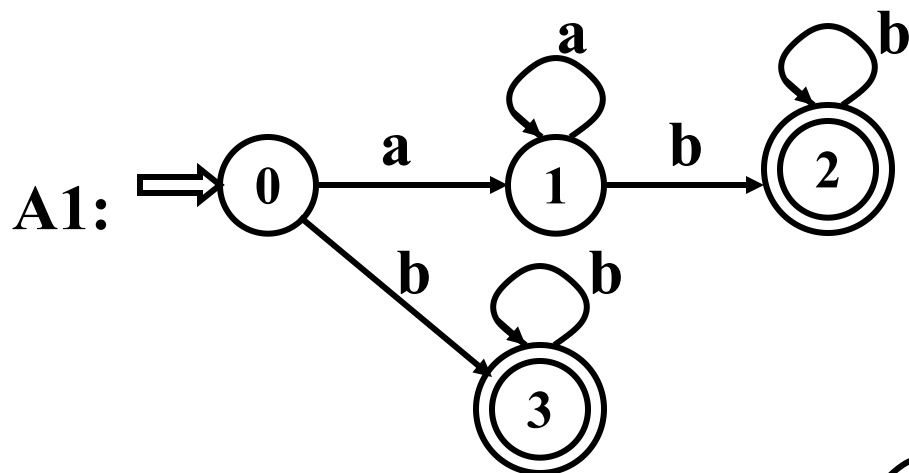
		Ia		Ib	
{i,1,2}	S	{1,2,3}	A	{1,2,4}	B
{1,2,3}	A	{1,2,3,5,6,f}	C	{1,2,4}	B
{1,2,4}	B	{1,2,3}	A	{1,2,4,5,6,f}	D
{1,2,3,5,6,f}	C	{1,2,3,5,6,f}	C	{1,2,4,6,f}	E
{1,2,4,5,6,f}	D	{1,2,3,6,f}	F	{1,2,4,5,6,f}	D
{1,2,4,6,f}	E	{1,2,3,6,f}	F	{1,2,4,5,6,f}	D
{1,2,3,6,f}	F	{1,2,3,5,6,f}	C	{1,2,4,6,f}	E

等价的**DFA**

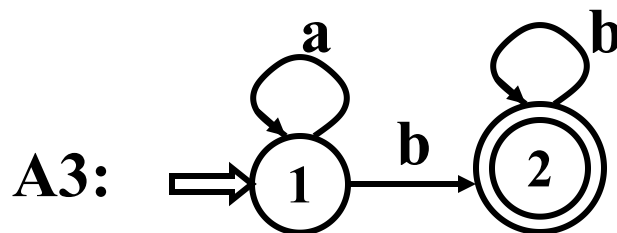


确定有穷自动机的化简

⌘ 化简就是对给定的确定自动机A1，构造另一个确定的自动机A2，使 $L(A2)=L(A1)$ 。并且A2的状态个数不多于A1的状态个数。



$$L(A1)=L(A2)=L(A3)$$



确定有穷自动机的化简

- ✘ 说一个有穷自动机是化简了的，即是说，它没有多余状态并且它的状态中没有两个是互相等价的。一个有穷自动机可以通过消除多余状态和合并等价状态而转换成一个最小的与之等价的有穷自动机。
- ✘ 所谓有穷自动机的多余状态，是指这样的状态：从自动机的开始状态出发，任何输入串也不能到达的那个状态；或者从这个状态没有通路到达终态。

DFA的最小化就是寻求最小状态DFA

最小状态DFA的含义:

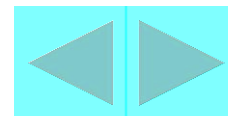
没有多余状态(死状态)

没有两个状态是互相等价 (不可区别)

两个状态s和t可区别: 不满足

兼容性——同是终态或同是非终态

传播性——从s出发读入某个a($a \in \Sigma$)和从t出发读入某个a到达的状态等价。



无关状态

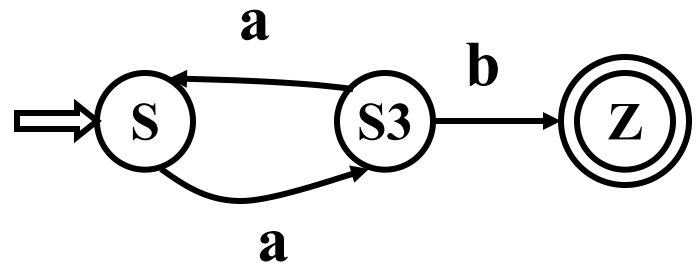
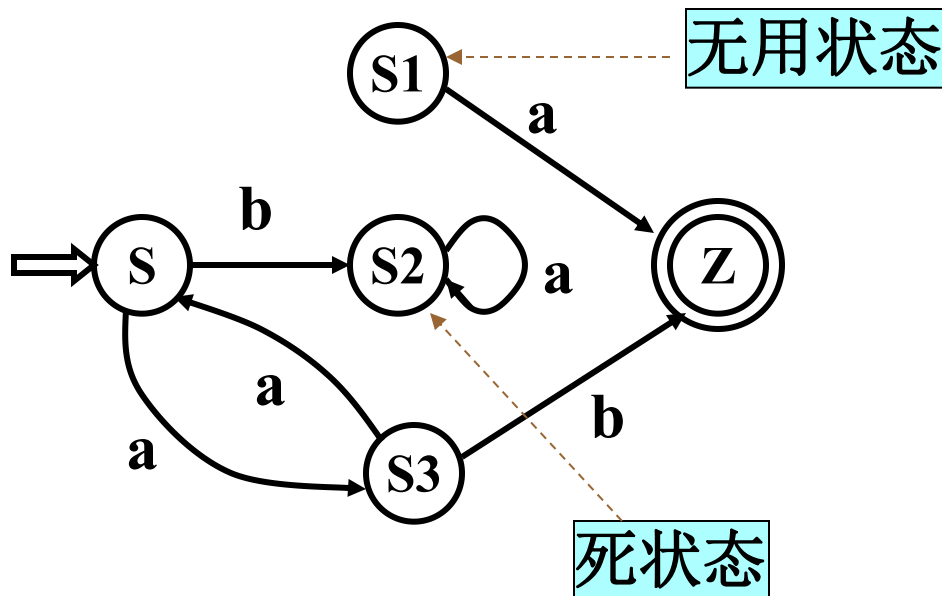
无用状态:

对于状态 S_i ，若从开始状态不能到达该状态，则称 S_i 为无用状态。

死状态:

对于状态 S_i ，若对任何输入符号 a 都不能从它到达终止状态，则称 S_i 为死状态。

无用状态和死状态统称为无关状态。



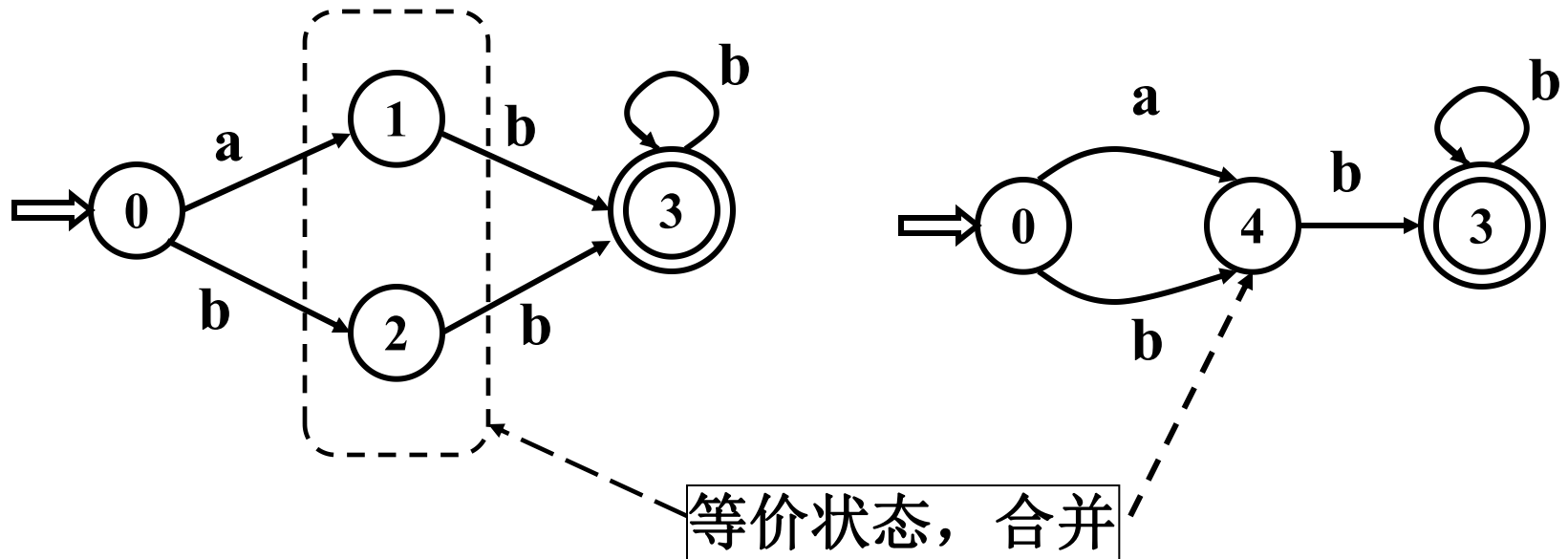
等价状态

等价状态：

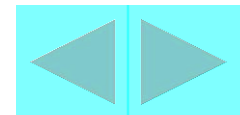
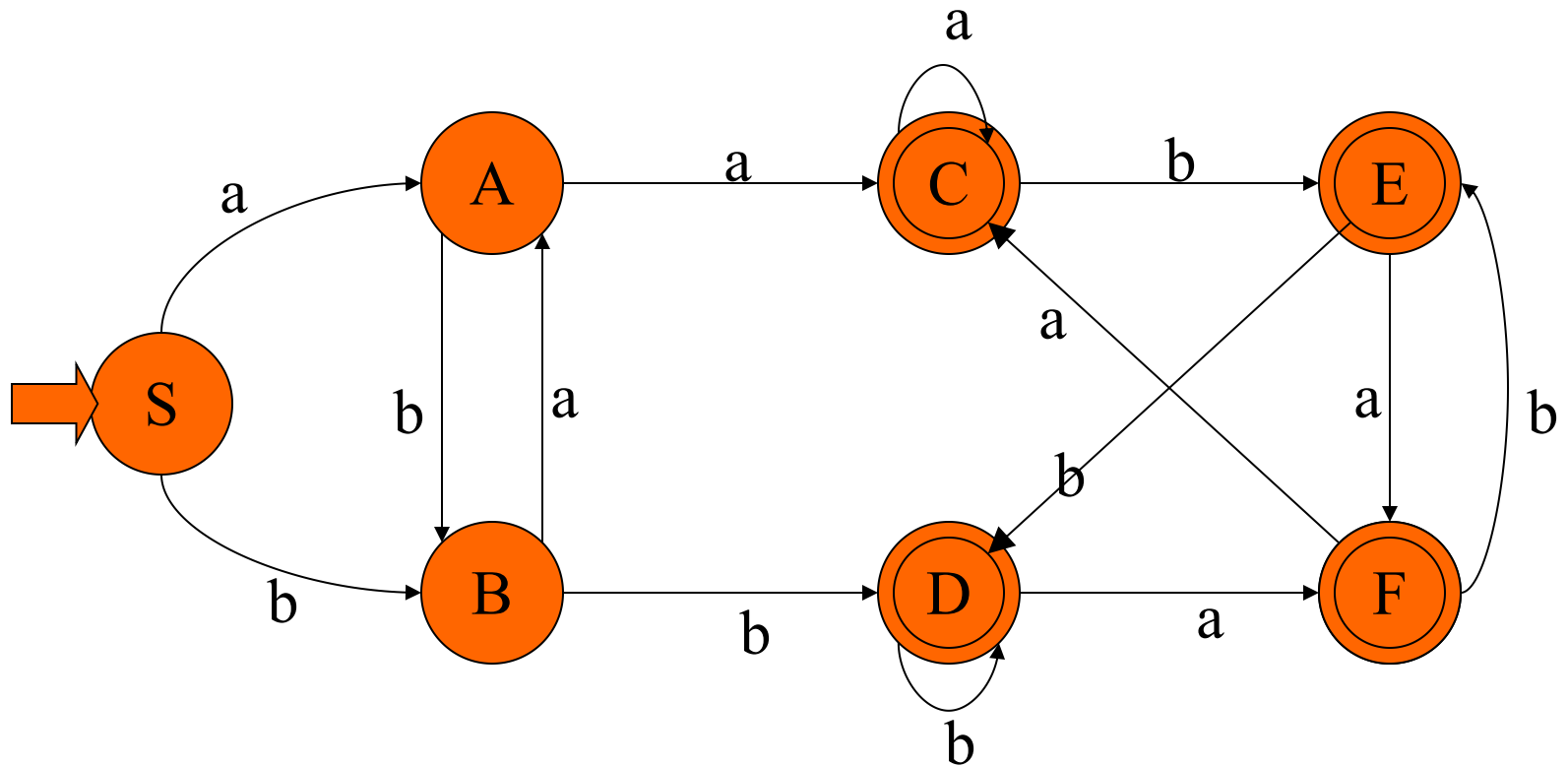
从 S_i 出发能导出的所有符号串集合记为 $L(S_i)$ ，设有两个状态 S_i 和 S_j ，若有

$$L(S_i) = L(S_j)$$

则称 S_i 和 S_j 是等价状态。



C和D同是终态,读入a到达C和F, C和F同是终态, C和F读入a都到达C,读入b都到达E. C和D等价



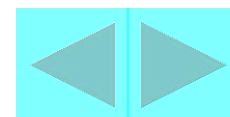
最小状态**DFA**

对于一个DFA $M = (K, \Sigma, f, k_0, k_t)$,
存在一个最小状态

DFA $M' = (K', \Sigma, f', k_0', k_t')$, 使 $L(M') = L(M)$.

结论

☑ 接受L的最小状态有穷自动机不计同构是唯一的。



“分割法”



DFA的最小化算法的核心

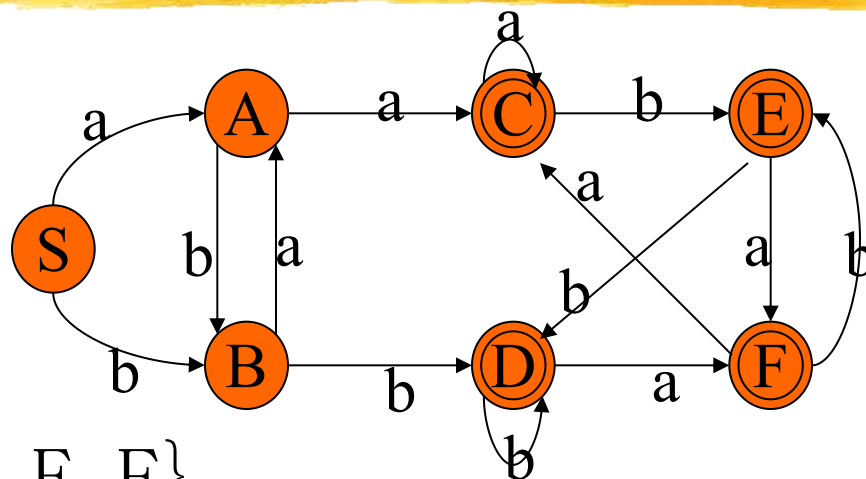
把一个**DFA**的状态分成一些不相交的子集，使得任何不同的两子集的状态都是可区别的，而同一子集中的任何两个状态都是等价的。

DFA的最小化—例子

$\Pi_0: \{S, A, B\}$

$\{C, D, E, F\}$

$\Pi_1: \{S, A, B\}$

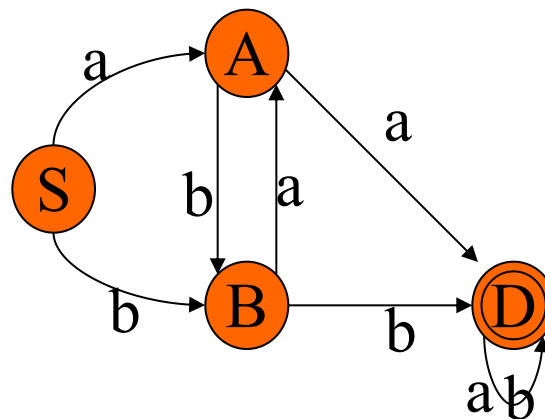


$\{A \mid S, B\}$

$\{C, D, E, F\}$

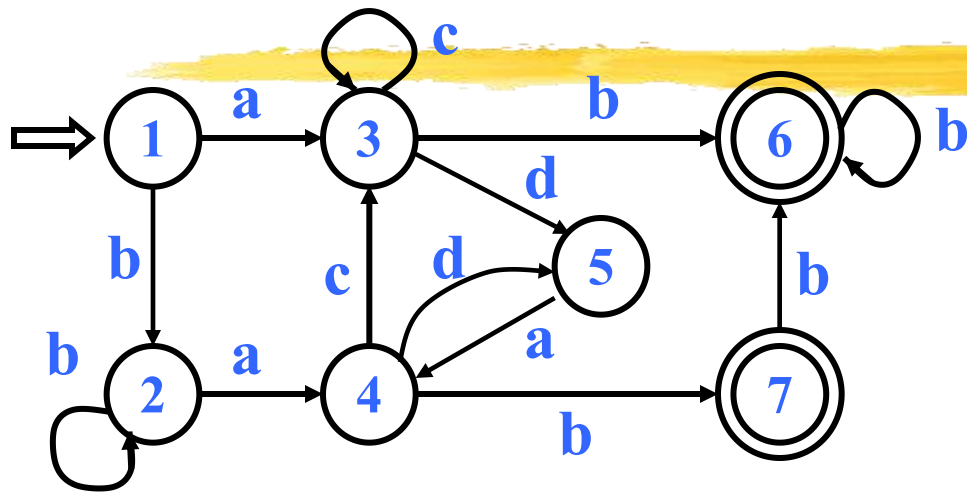
$\Pi_2:$

$\{S\} \mid \{B\}$



化简DFA的例子

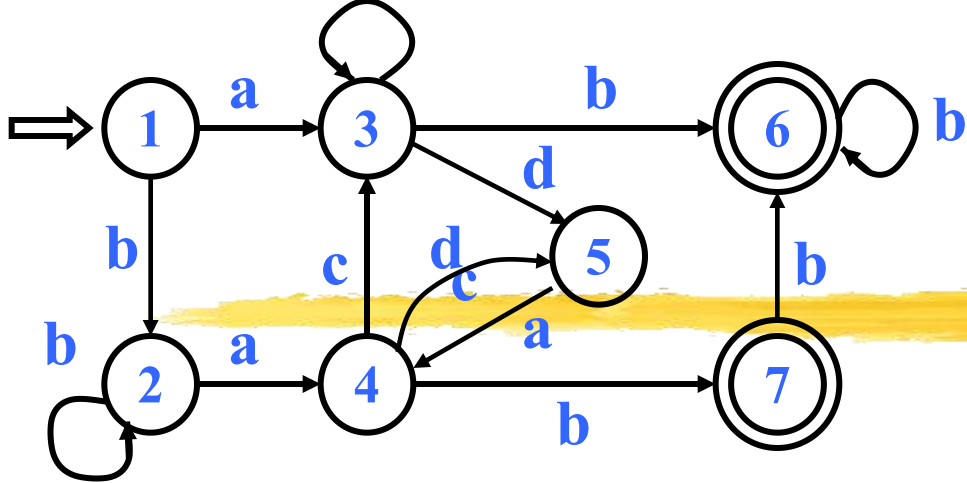
习题4.9 将下面的DFA最小化，并用正规式描述它所识别的语言。



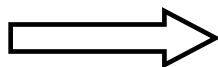
$\Pi 1$

	a	b	c	d
1	3	2		
2	4	2		
3		6	3	5
4		7	3	5
5	4			
6		6		
7		6		

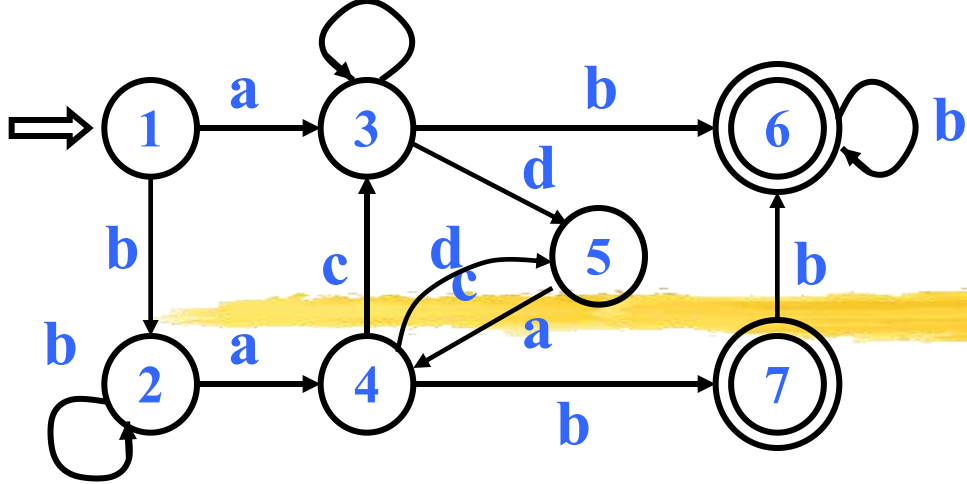
$\Pi 2$



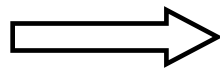
	a	b	c	d
Π_1 1	3	2		
2	4	2		
3		6	3	5
4		7	3	5
5	4			
Π_2 6		6		
7		6		



	a	b	c	d
Π_1 1	3	2		
2	4	2		
5	4			
Π_3 3		6	3	5
4		7	3	5
Π_2 6		6		
7		6		

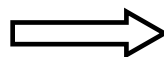


		a	b	c	d
Π_1	1	3	2		
	2	4	2		
	5	4			
Π_3	3		6	3	5
	4		7	3	5
Π_2	6		6		
	7		6		

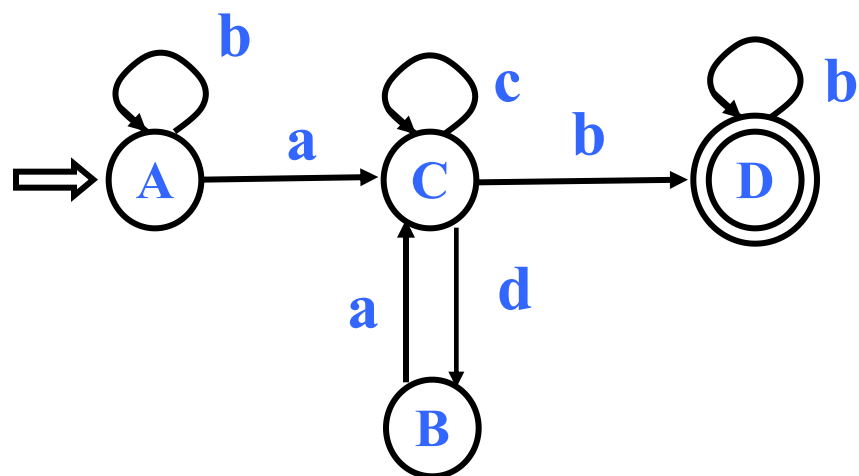


		a	b	c	d
Π_1	1	3	2		
	2	4	2		
Π_4	5	4			
Π_3	3		6	3	5
	4		7	3	5
Π_2	6		6		
	7		6		

		a	b	c	d
$\Pi 1$	1	3	2		
	2	4	2		
$\Pi 4$	5	4			
$\Pi 3$	3		6	3	5
	4		7	3	5
$\Pi 2$	6		6		
	7		6		



		a	b	c	d
	A	C	A		
	B	C			
	C		D	C	B
	D		D		



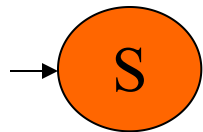
$b^*a(c|da)^*bb^*$

词法分析程序的自动构造

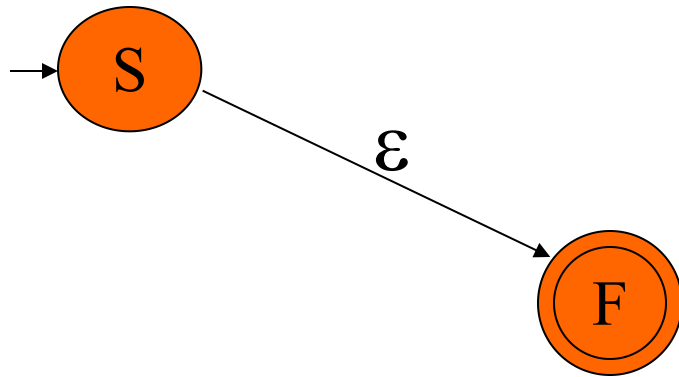
对有穷自动机和正规表达式进行了上述讨论之后,我们介绍词法分析程序的自动构造方法,这个方法基于有穷自动机和正规表达式的等价性,即:

1. 对于 Σ 上的一个 NFA M , 可以构造一个 Σ 上的正规式 R , 使得 $L(R)=L(M)$ 。
2. 对于 Σ 上的一个正规式 R , 可以构造一个 Σ 上的 NFA M , 使得 $L(M)=L(R)$ 。

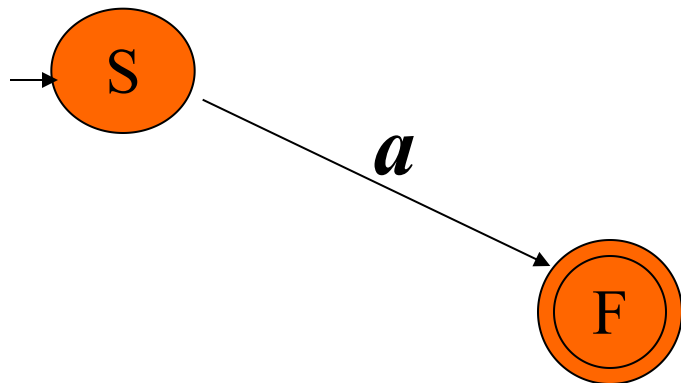
对于正规式 $\mathbf{R}=\emptyset$,构造的 **NFA**



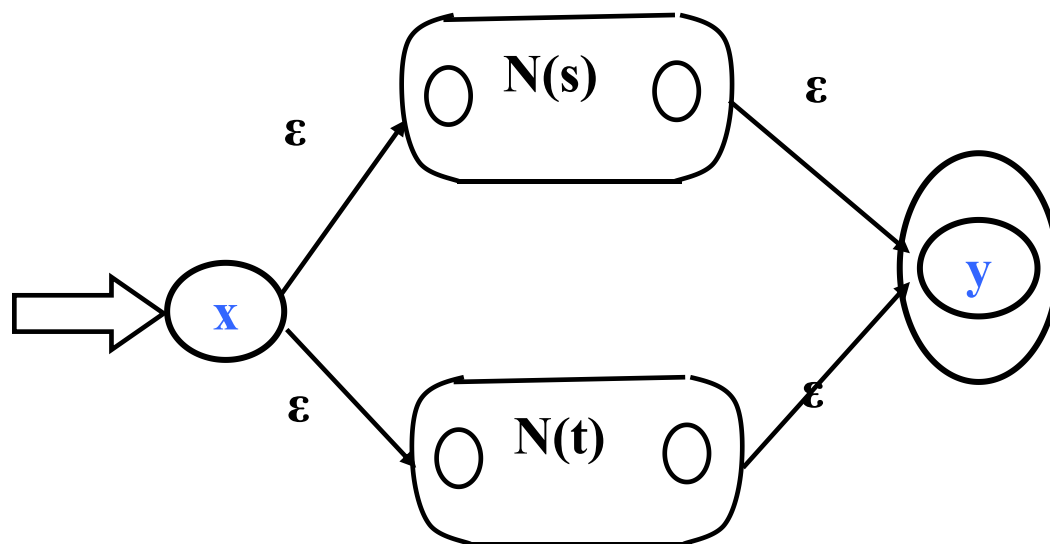
对于正规式 **$R = \varepsilon$** , 构造的 **NFA**



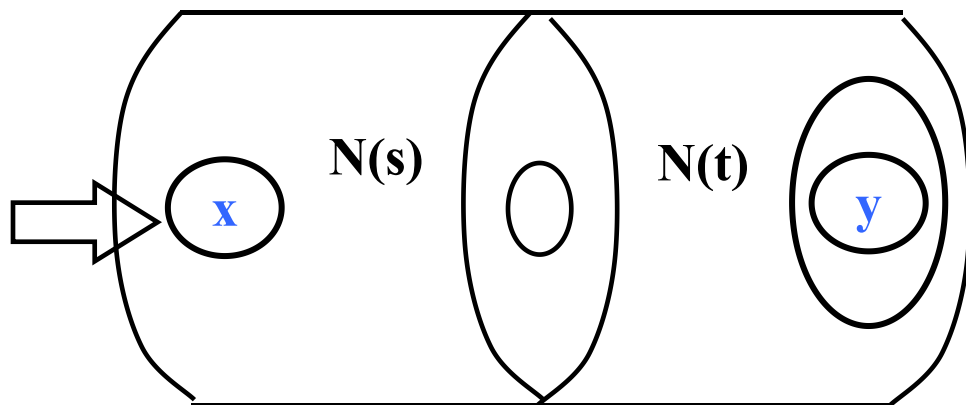
对于正规式 **$R=a$** ,构造的**NFA**



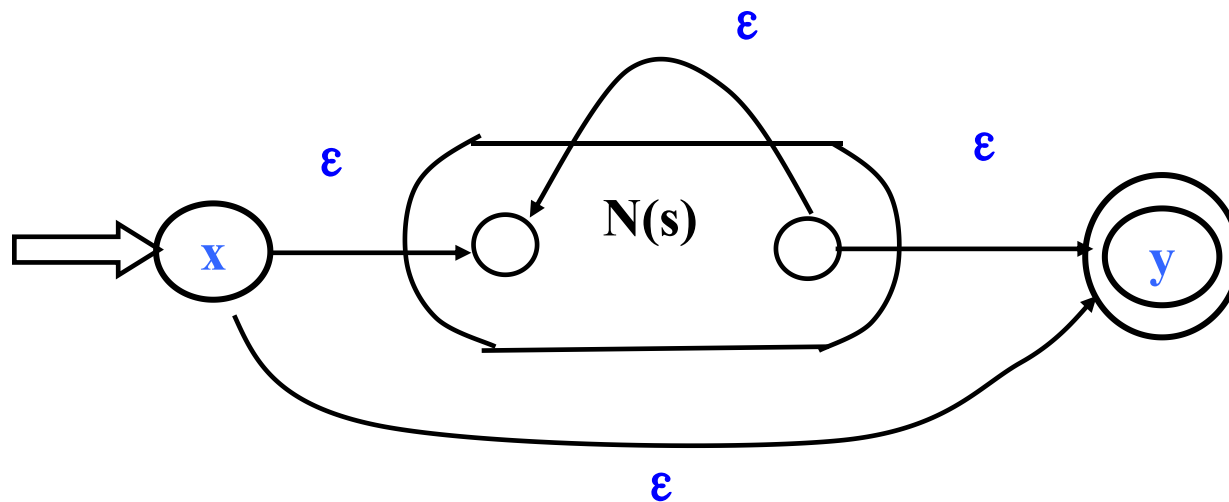
对于正规式 r , $r = s|t$ 构造的NFA



对于正规式 r , $r=st$ 构造的**NFA**



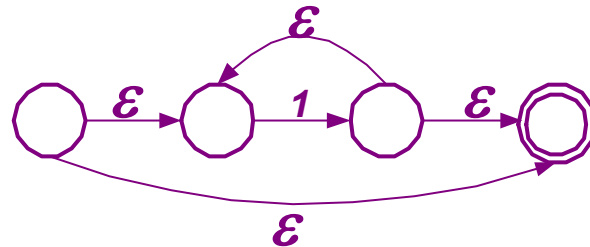
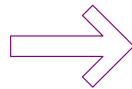
对于正规式 r , $r=s^*$ 构造的**NFA**



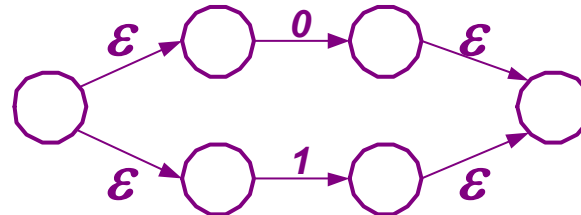
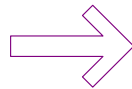
举例:从正规表达式构造等价的 ε -NFA

正规表达式 $1^*0(0|1)^*$

1^*

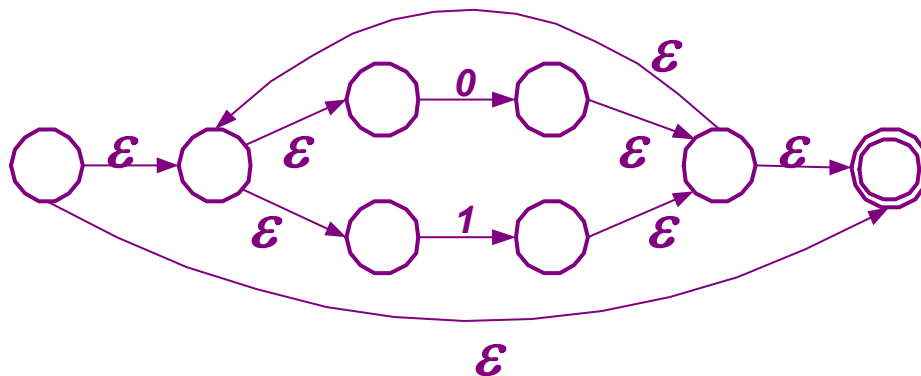
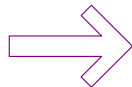


$0|1$

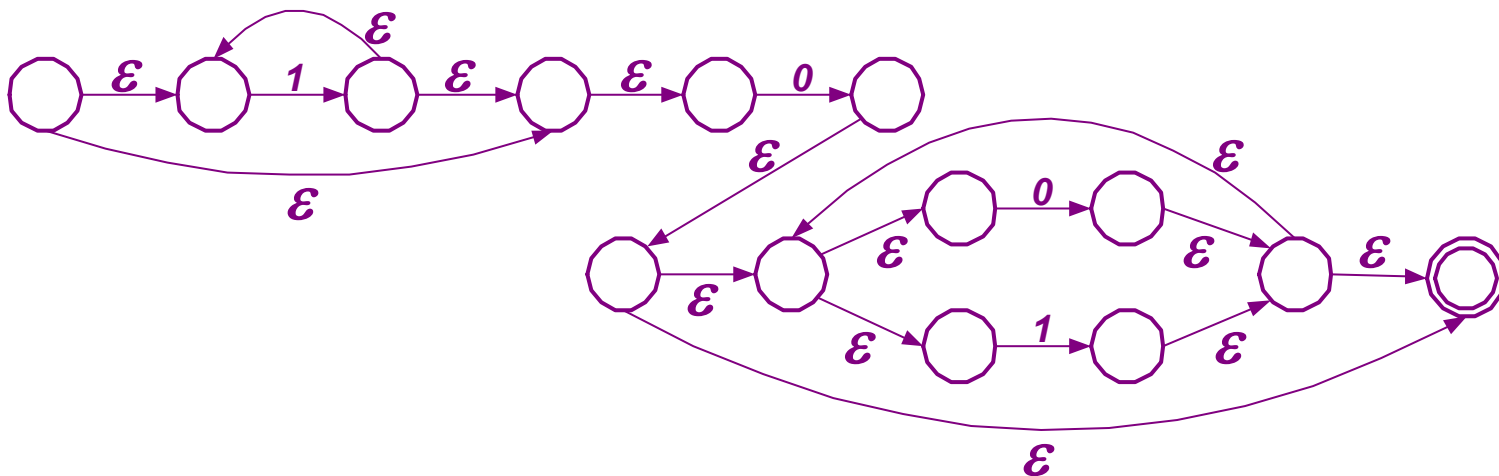


从正规表达式构造等价的 ε -NFA

$(0|1)^*$

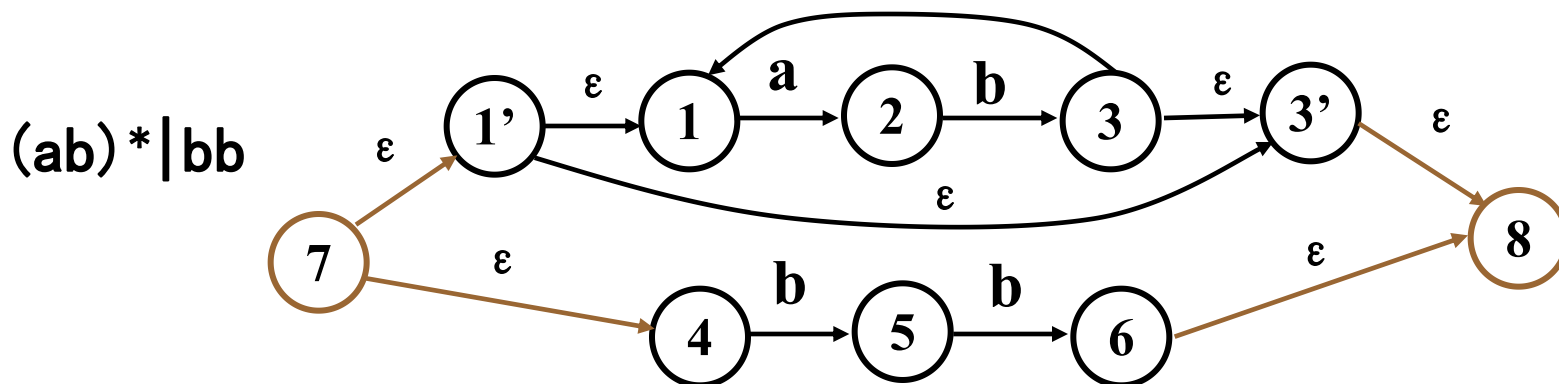
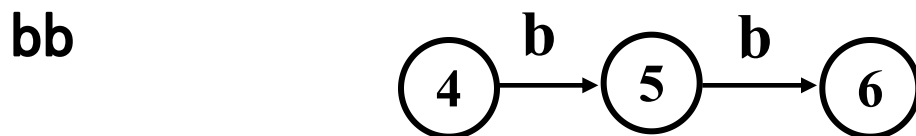
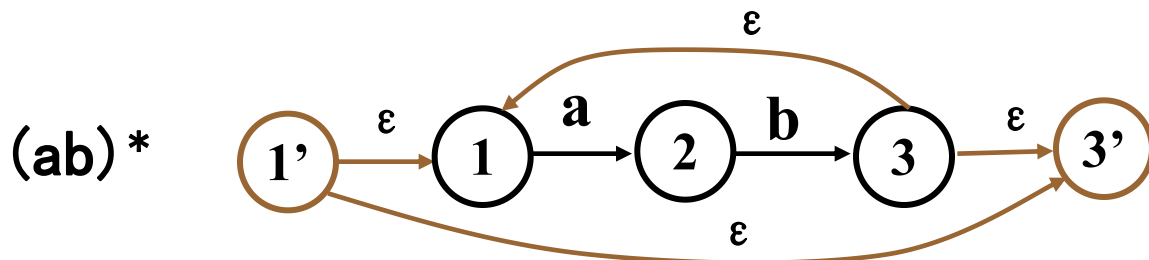
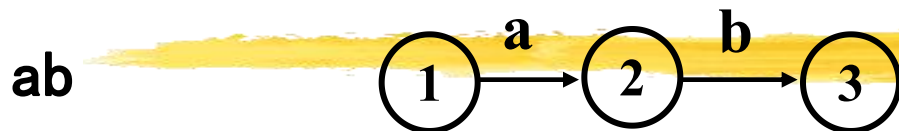


$1^*0(0|1)^*$



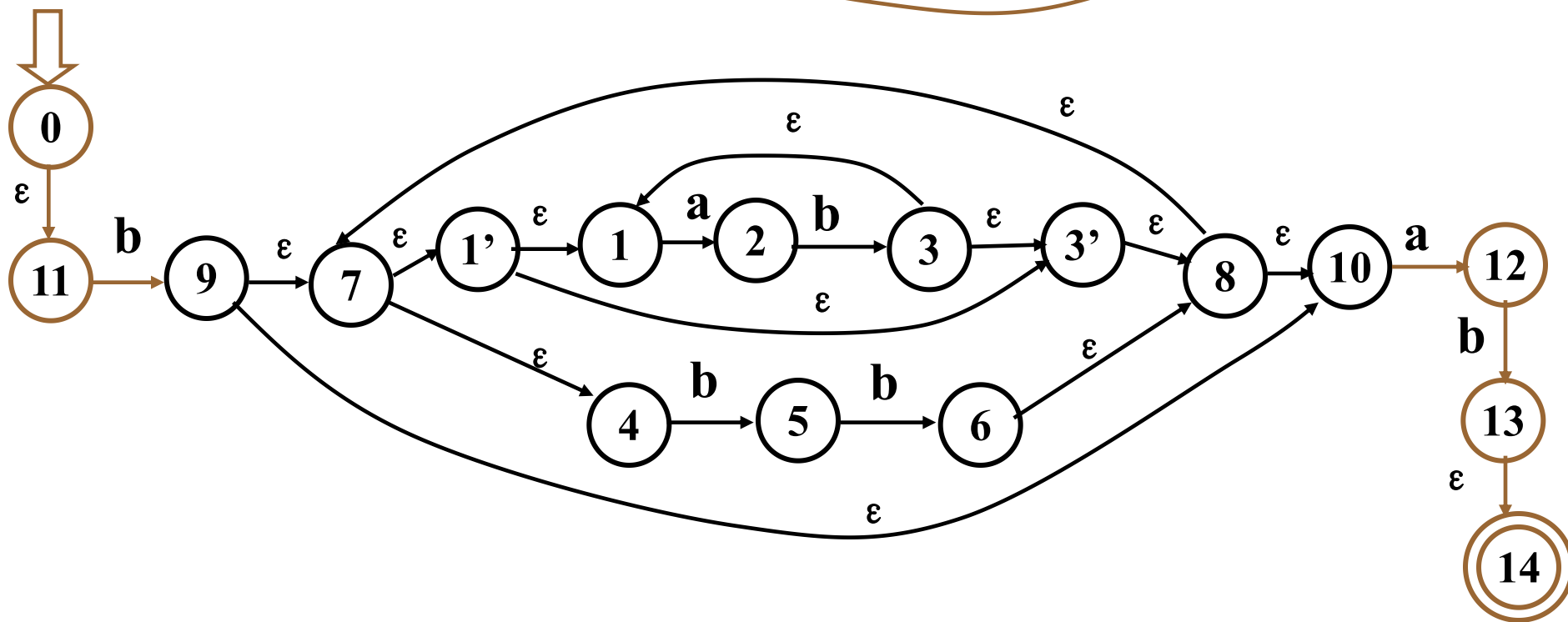
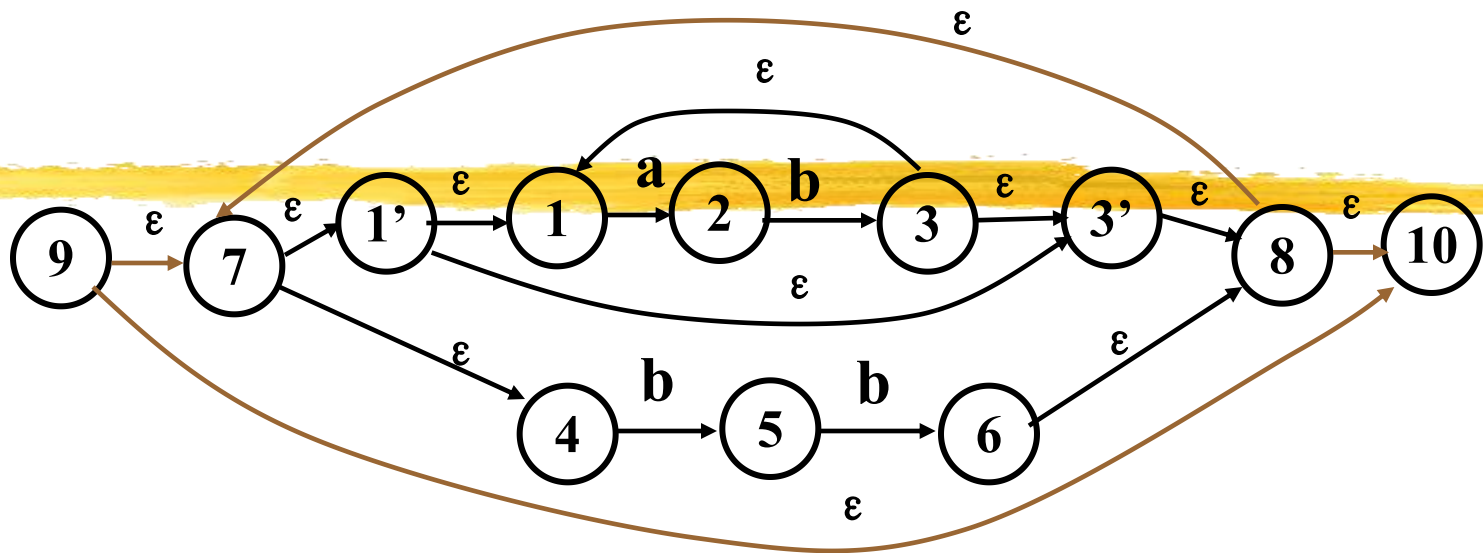
例、由正规式构造NFA

例、构造正规式 $r = b((ab)^* | bb)^* ab$ 的NFA

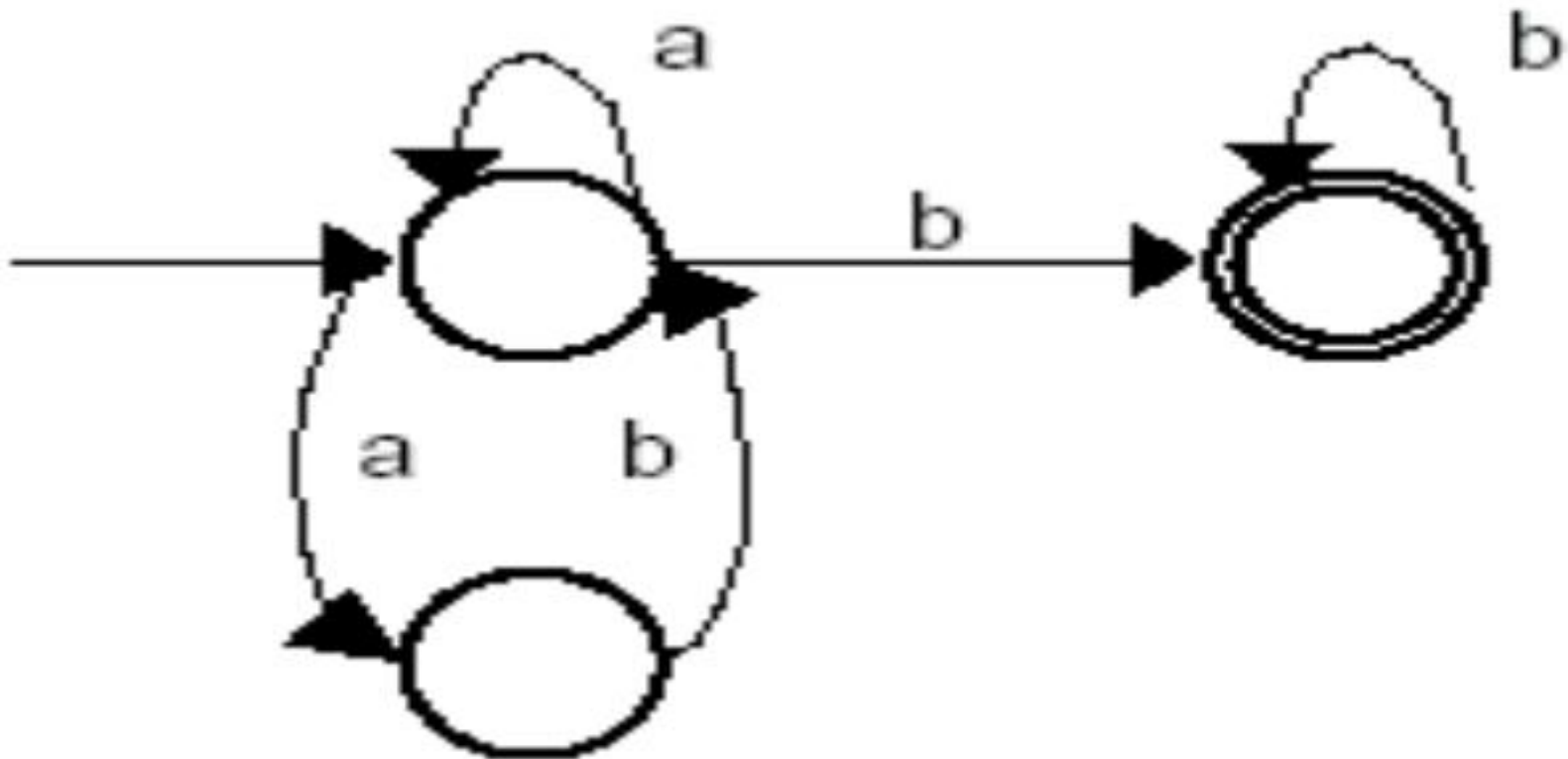



例、构造正规式 $r=b((ab)^*|bb)^*ab$ 的NFA

$((ab)^*|bb)^*$




$$R = (a|ab)^* b b^*$$





正规式用于说明(描述)单词的结构十分简洁方便。而把一个正规式编译(或称转换)为一个**NFA**进而转换为相应的**DFA**，这个**NFA**或**DFA**正是识别该正规式所表示的语言的句子的识别器。基于这种方法来构造词法分析程序



词法分析程序的设计技术可应用于其它领域，比如查询语言以及信息检索系统等，这种应用领域的程序设计特点是，通过字符串模式的匹配来引发动作

又如**LEX**，说明词法分析程序的语言，可以看成是一个模式动作语言。

词法分析程序的自动构造工具也广泛应用于许多方面，如用以生成一个程序，可识别印刷电路板中的缺陷，又如开关线路设计和文本编辑的自动生成等。

本章小结



词法分析程序是编译第一阶段的工作，它读入字符流的源程序，按照词法规则识别单词，交由语法分析程序接下去。

本章讲述了词法分析程序设计原则，并介绍了分别作为正规集描述和识别机制的正规式和有穷动机。在此基础上给出了词法分析程序自动构造工具如**LEX**的原理。

识别P10单词的DFA

