

第3章 词法分析

Part II 自动机

主要内容

- DFA
- NFA
- $\text{NFA} \Rightarrow \text{DFA}$

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, 即识别正规文法所定义的语言和正规式所表示的集合, 引入有穷自动机这个理论, 正是为词法分析程序的自动构造寻找特殊的方法和工具

有穷自动机分为两类:

确定的有穷自动机(Deterministic Finite Automata)

不确定的有穷自动机(Nondeterministic Finite Automata)

确定的有穷自动机

Deterministic Finite Automata

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: $M = (K, \Sigma, f, S, Z)$, 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 为转换函数，是在 $K \times \Sigma \rightarrow K$ 上的映射，即，如 $f(k_i, a) = k_j$, ($k_i \in K, k_j \in K$) 意味着，如果当前状态为 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:

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

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

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

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

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

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

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

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

一个DFA可以表示成一个**状态图**(或称状态转换图)

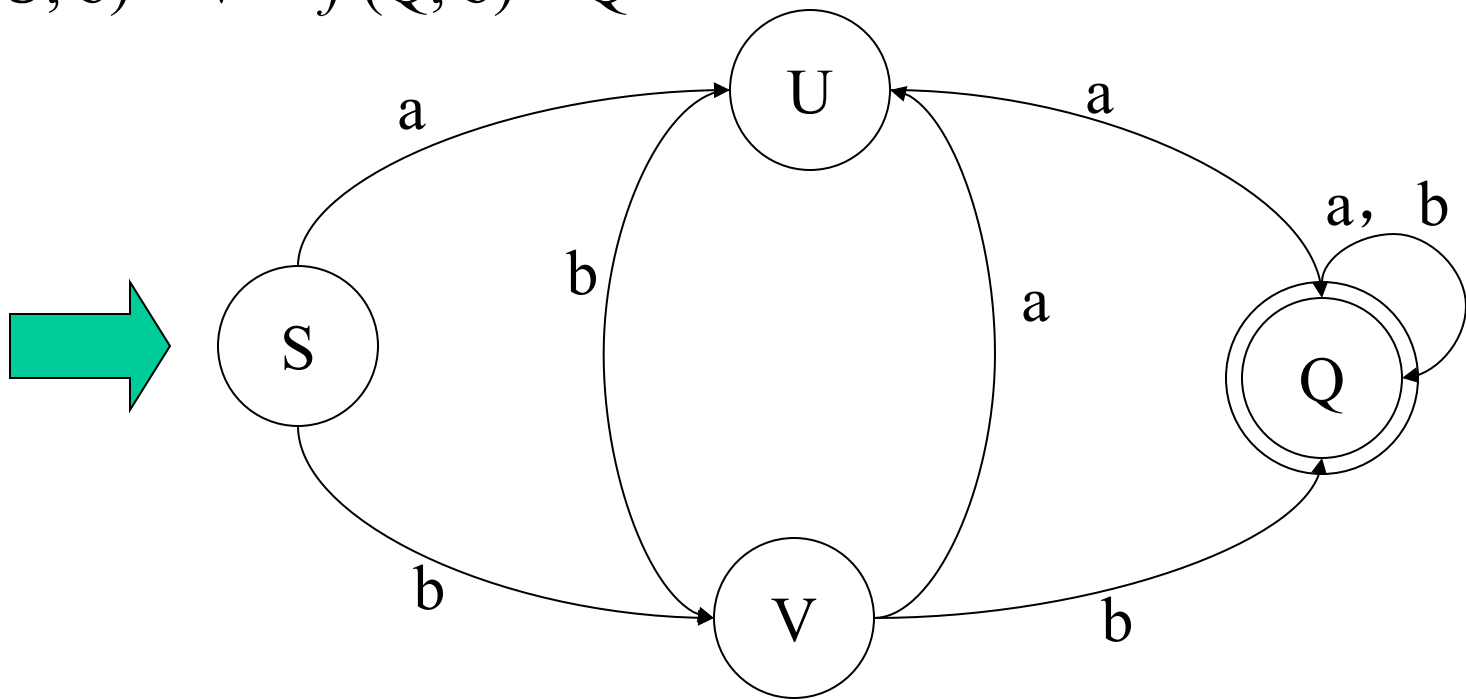
假定DFA M 含有 m 个状态, n 个输入字符, 那么这个状态图含有 m 个结点, 每个结点最多有 n 个弧射出;

整个图含有唯一一个初态结点和若干个终态结点, 初态结点冠以双箭头 " \Rightarrow " 或标以 " $-$ ", 终态结点用双圈表示或标以 " $+$ ";

若 $f(k_i, a)=k_j$, 则从状态结点 k_i 到状态结点 k_j 画标记为 a 的弧;

DFA 的状态图表示

$$\begin{array}{ll} f(S, a) = U & f(V, a) = U \\ f(S, b) = V & f(V, b) = Q \\ f(U, a) = Q & f(Q, a) = Q \\ f(U, b) = V & f(Q, b) = Q \end{array}$$



一个DFA还可以用一个矩阵表示:

该矩阵的行表示状态，列表示输入字符，
矩阵元素表示相应状态行和输入字符列下的新状态，即k行a列为 $f(k, a)$ 的值。

用双箭头 " \Rightarrow " 标明初态；否则第一行即是初态，相应终态行在表的右端标以1，非终态标以0。

DFA 的矩阵表示

状态 \ 字符	a	b	
S	U	V	0
U	Q	V	0
V	U	Q	0
Q	Q	Q	1

DFA作为一种识别机制

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：

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

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:

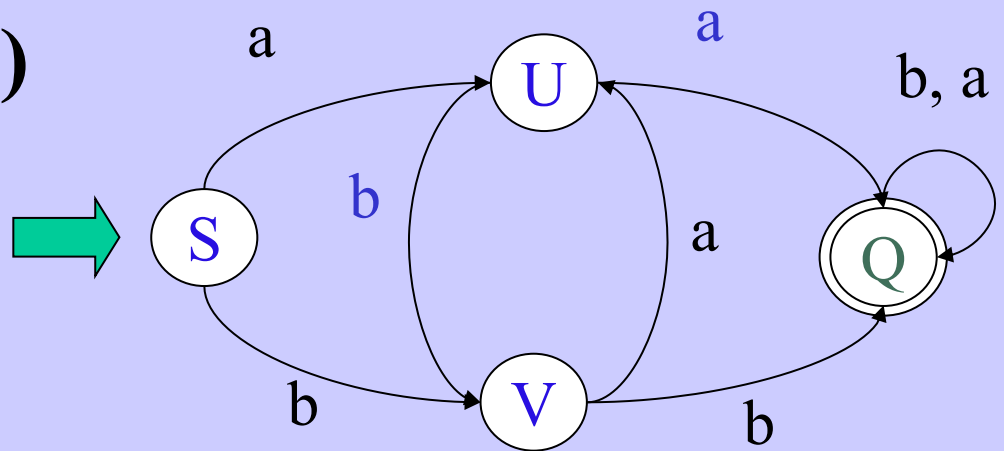
$M=(K, \Sigma, f, S, Z)$,

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)

例：证明 $t=baab$ 被下图的DFA所接受。

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



DFA M 所能接受的符号串的全体记为 $L(M)$

对于任何两个有穷自动机 M 和 M' ，如果
 $L(M)=L(M')$ ，则称 M 与 M' 是等价的

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，那么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

用程序来模拟DFA的行为:

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

$K := S;$

$c := \text{getchar};$

while $c \neq \text{eof}$ *do*

{

$K := f(K, c);$

$c := \text{getchar};$

};

if K is in Z *then return* ('yes')

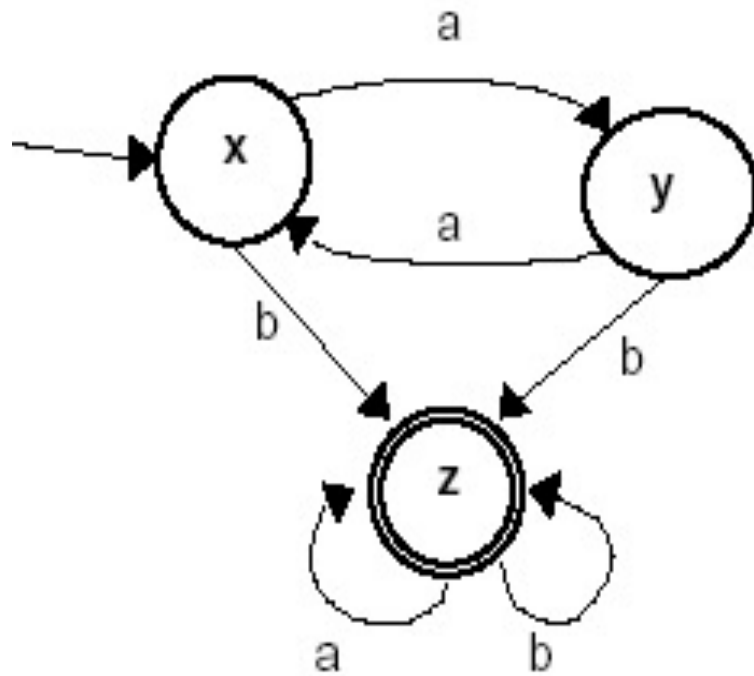
else return ('no')

Review

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

- 1) A finite set of states, one of which is designated the initial state or *start state*, and some of which are designated as final states.
- 2) An alphabet of possible input symbols.
- 3) A finite set of transitions that specifies for each state and for each symbol of the input alphabet, which state to go to next.

DFA examples

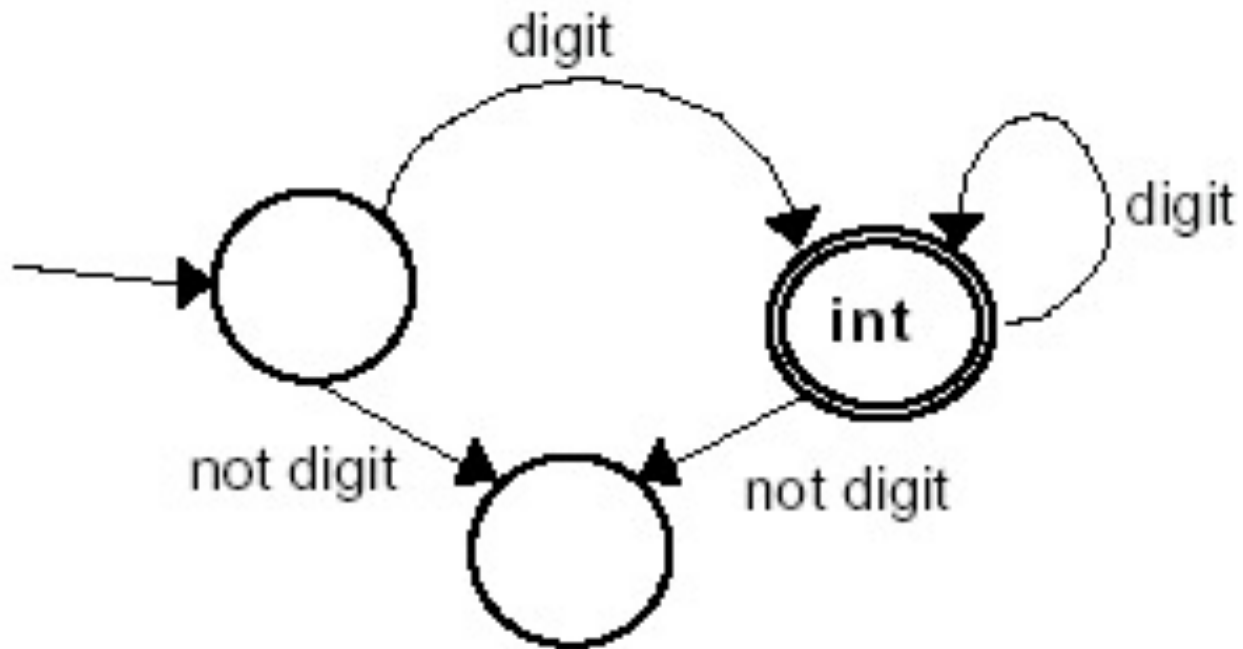


(start)

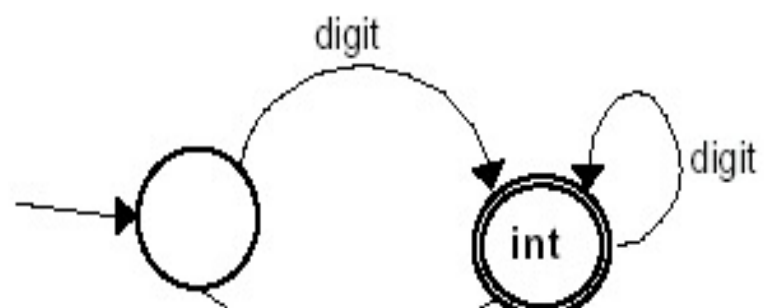
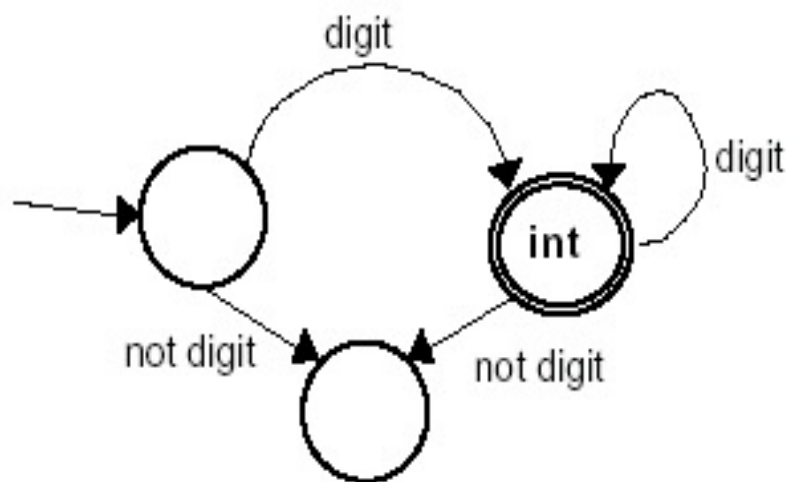
(final)

	a	b
x:	y	z
y:	x	z
z:	z	z

DFA examples



FA 等价



不确定的有穷自动机

Nondeterministic Finite Automata

有穷自动机

有穷自动机（即有限自动机）是一种**形式系统**，也是一种**识别装置**，能识别**正规集**，即正规文法所定义的语言和正规式所表示的集合

有穷自动机分类

确定的有穷自动机（Deterministic Finite Automata）

不确定的有穷自动机（Nondeterministic Finite Automata）

确定的有穷自动机DFA：回顾

- 一个确定的有穷自动机DFA M 是一个五元组： $M = (K, \Sigma, f, S, Z)$ ，其中
 - K 是一个有穷集，它的每个元素称为一个状态；
 - 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；
 - f 为转换函数，是 $K \times \Sigma \rightarrow K$ 上的映射， $f(k_i, a) = k_j$ ($k_i \in K, k_j \in K$) 意味着，如果当前状态为 k_i ，输入符号为 a 时，将转换为下一个状态 k_j ；
 - $S \in K$ 是唯一的初态；
 - $Z \subseteq K$ 是一个终态集，终态也称可接受状态或结束状态。

对比DFA：NFA不确定性体现在哪？

DFA	NFA
有穷状态集 K	✓
有穷字母表 Σ	✓
转换函数 $f: K \times \Sigma \rightarrow K$	✗
唯一初态 $S \in K$?
终态集 $Z \subseteq K$	✓

DFA转换函数：如何引入不确定性？

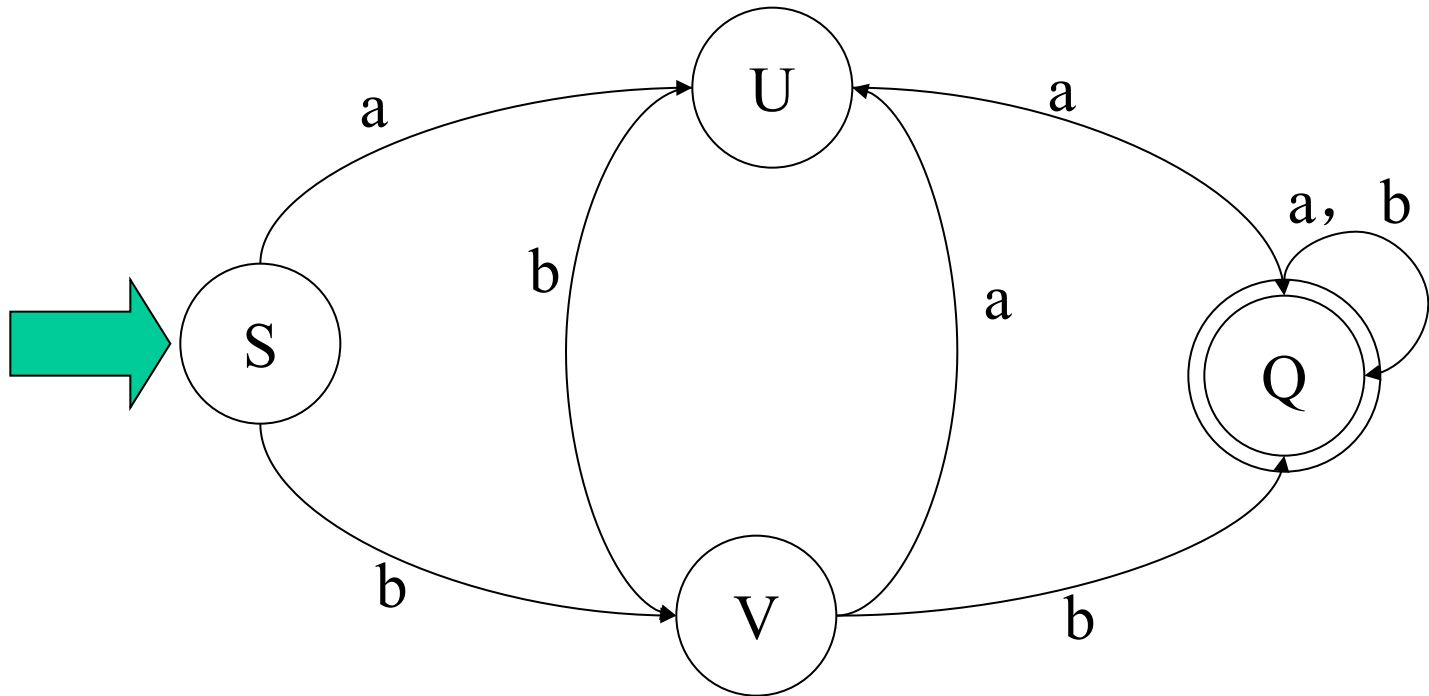
转换函数 $f: K \times \Sigma \rightarrow K$

$$f(S, a) = U \quad f(V, a) = U$$

$$f(S, b) = V \quad f(V, b) = Q$$

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

$$f(U, b) = V \quad f(Q, b) = Q$$



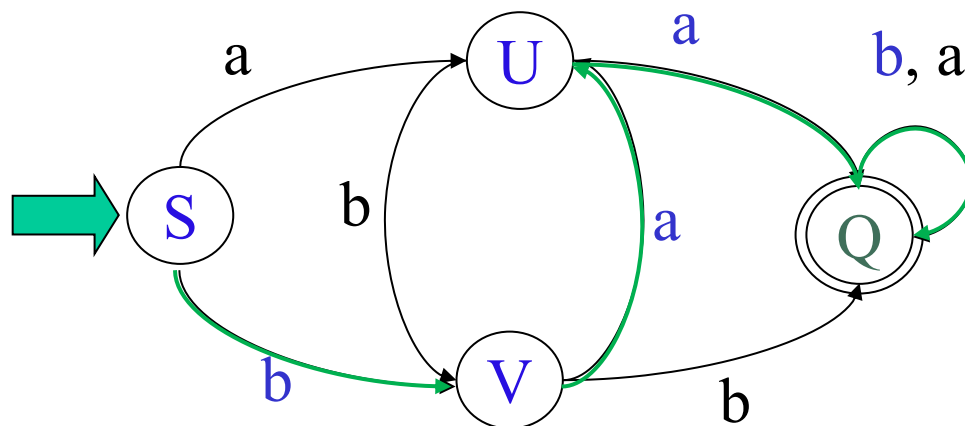
DFA识别机制：如何引入不确定性？

例：证明 $t=baab$ 被下图所示DFA所接受

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

扩充的转换函数

$$f: K \times \Sigma^* \rightarrow K$$

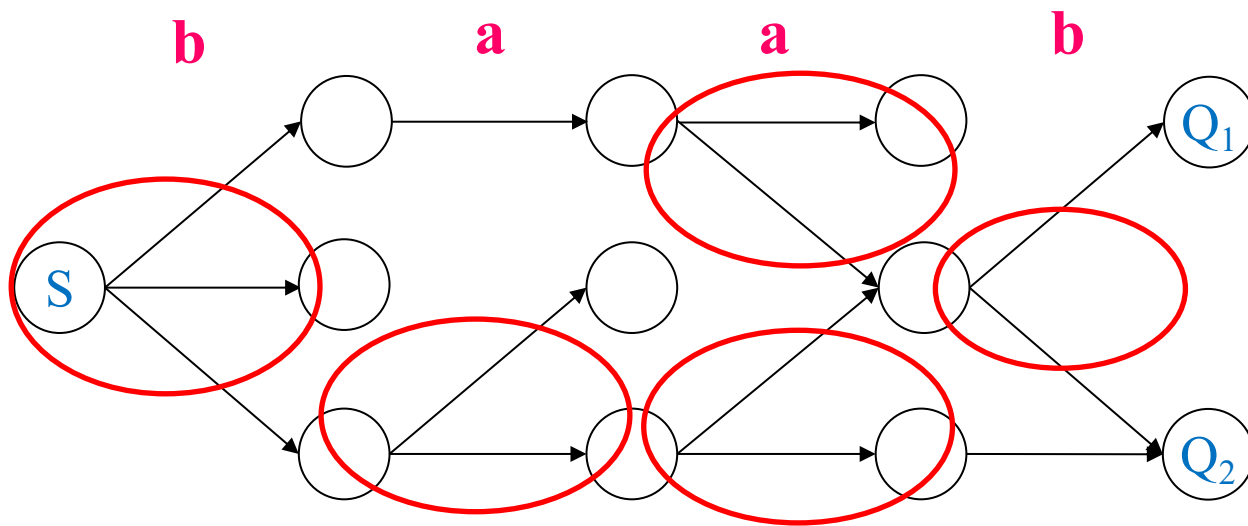


确定性：唯一路径

↓
不确定性：不唯一路径

NFA转换函数：如何刻画多条路径？

例：字符串 $t = \text{baab}$ 的多条识别路径



DFA

$f(S, a) = U$ $f(U, a) = Q$
 $f(S, b) = V$ $f(U, b) = V$
 $f(V, a) = U$ $f(Q, a) = Q$
 $f(V, b) = Q$ $f(Q, b) = Q$



NFA

$f(S, a) = \{U, V\}$ $f(U, a) = Q$
 $f(S, b) = V$ $f(U, b) = V$
 $f(V, a) = U$ $f(Q, a) = Q$
 $f(V, b) = \{U, Q\}$ $f(Q, b) = Q$

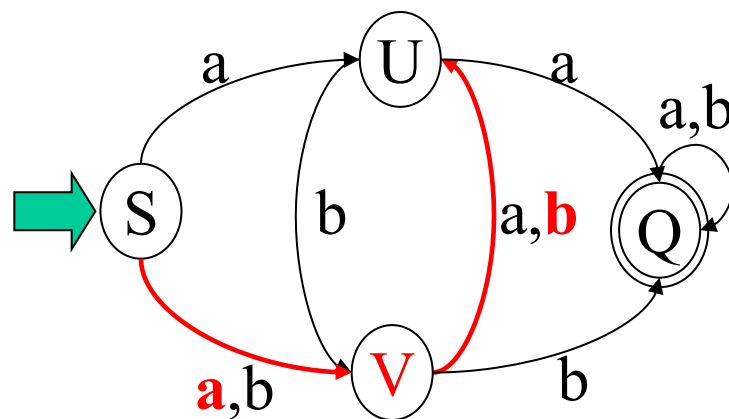
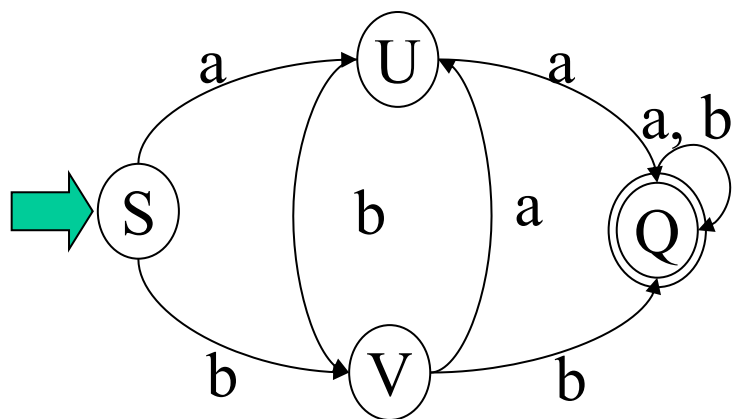
NFA转换函数： $f: K \times \Sigma \rightarrow 2^K$

DFA转换函数 \rightarrow NFA转换函数

$$\begin{array}{ll} f(S, a) = U & f(U, a) = Q \\ f(S, b) = V & f(U, b) = V \\ f(V, a) = U & f(Q, a) = Q \\ f(V, b) = Q & f(Q, b) = Q \end{array}$$



$$\begin{array}{ll} f(S, a) = \{U, V\} & f(U, a) = Q \\ f(S, b) = V & f(U, b) = V \\ f(V, a) = U & f(Q, a) = Q \\ f(V, b) = \{U, Q\} & f(Q, b) = Q \end{array}$$



转换函数 $f: K \times \Sigma \rightarrow K$



转换函数 $f: K \times \Sigma \rightarrow 2^K$

不确定的有穷自动机NFA：定义

- 一个不确定的有穷自动机NFA M 是一个五元组 $M = (K, \Sigma, f, S, Z)$ ，其中
 - K 是一个有穷集，它的每个元素称为一个状态；
 - 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；
 - f 为转换函数，是 $K \times \Sigma \rightarrow 2^K$ 上的映射， $f(k_i, a) = \{k_{j1}, \dots, k_{jn}\} \subseteq K$ ，如果当前状态为 k_i ，输入符号为 a 时，将转换为 $\{k_{j1}, \dots, k_{jn}\}$ 中的某个状态；
 - $S \subseteq K$ 是初态集；
 - $Z \subseteq K$ 是一个终态集，终态也称可接受状态或结束状态。

NFA $M=(K, \Sigma, f, S, Z)$ 的识别机制

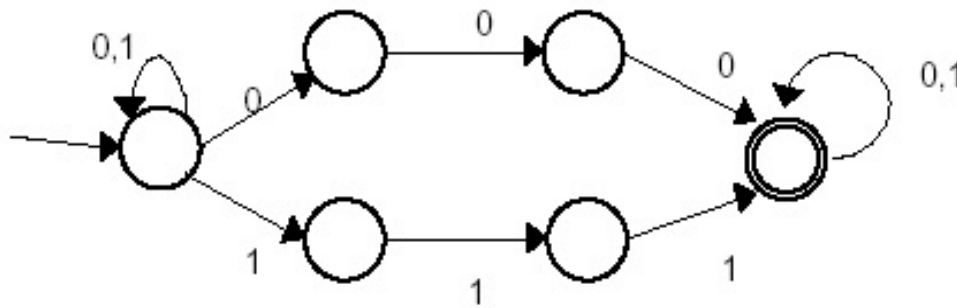
- Σ^* 上的符号串 t 在 NFA M 上运行
 - 一个输入符号串 $t = Tt_1$ ，其中 $T \in \Sigma$ ， $t_1 \in \Sigma^*$ （转换函数扩充为 $K \times \Sigma^* \rightarrow 2^K$ ），则
$$f(Q, Tt_1) = f(f(Q, T), t_1), \text{ 其中 } Q \in K$$
- Σ^* 上的符号串 t 被 NFA M 接受（识别）
 - 若 $t \in \Sigma^*$ ， $f(S_0, t) = P$ ， $S_0 \in S$ ，则
$$P \cap Z \neq \emptyset$$

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, 当且仅当存在一个 Σ 上的不确定的有穷自动机 M , 使得 $V = L(M)$

NFA 示例



000

1010001

11

011001

例子：

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

其中，

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

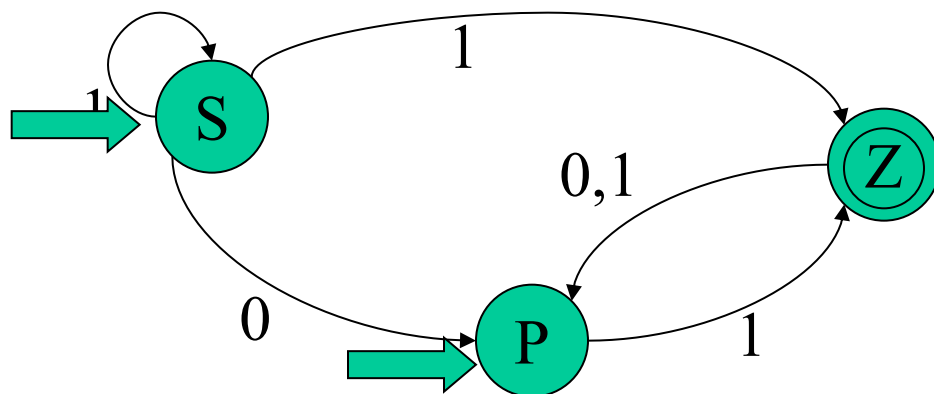
$$f(Z, 0) = \{P\}$$

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

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

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

状态图表示



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

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

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

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

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

矩阵表示

<div></div>	0	1	
S	{P}	{S,Z}	0
P	{}	{Z}	0
Z	{P}	{P}	1

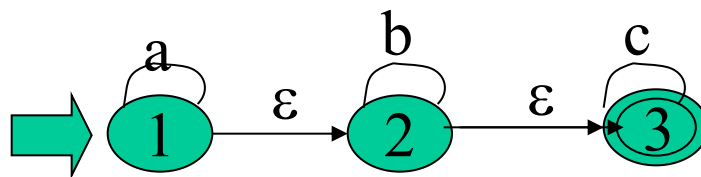
简化为



<div></div>	0	1	
S	P	S,Z	0
P	.	Z	0
Z	P	P	1

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

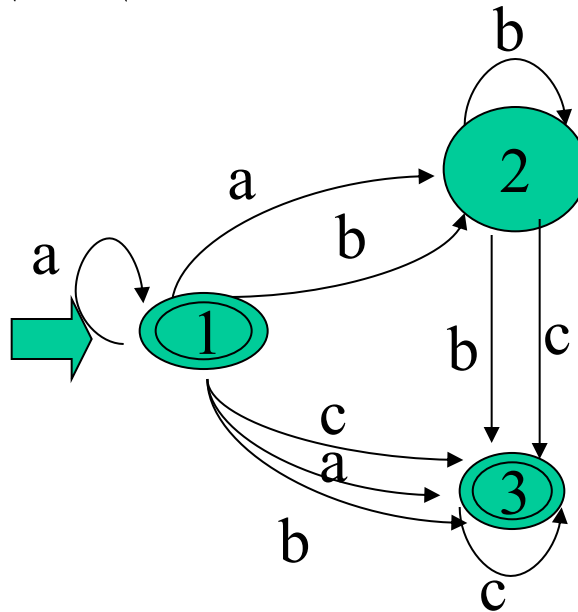
具有 ϵ 转移的不确定有穷自动机



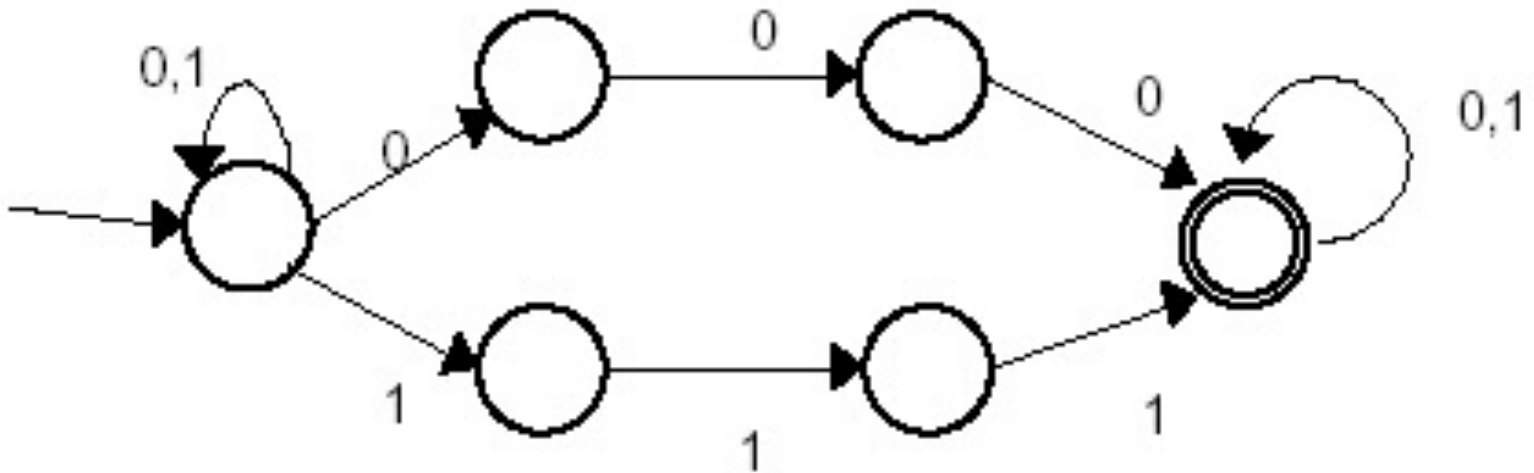
定理

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:



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



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

有一种算法，可以将NFA转换成接受同样语言的DFA。这种算法称为**子集法**

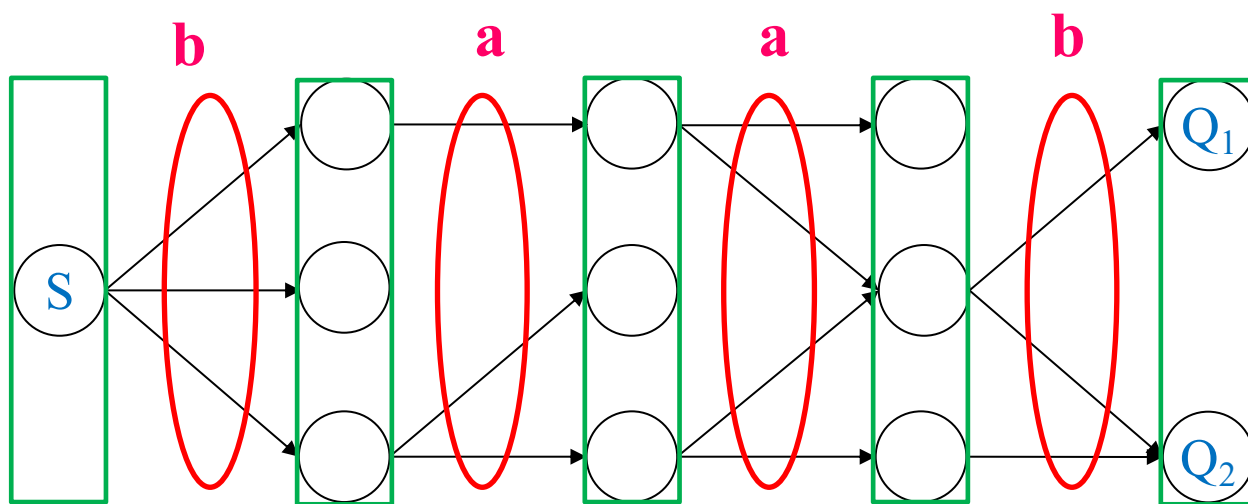
与某一NFA等价的DFA不唯一

NFA \Rightarrow DFA

子集法

NFA=>DFA, 如何确定地刻画多条路径?

例：字符串 $t = \mathbf{baab}$ 的多条识别路径



从**NFA**的矩阵表示中可以看出，表项通常是一状态的集合；而在**DFA**的矩阵表示中，表项是一个状态；

因此，从**NFA**到**DFA**的基本构造思路是：

DFA的每一个状态对应NFA的一组状态

DFA使用它的状态去记录在NFA读入一个输入符号后可能达到的所有状态

NFA确定化算法

NFA $N=(K, \Sigma, f, K_0, K_t)$, 按如下方法构造一个 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，即为 Σ ；
3. 转换函数是这样定义的：

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

$$\{R_1, R_2, \dots, R_t\} = \varepsilon\text{-closure}(\text{move}(\{S_1, S_2, \dots, S_j\}, a))$$
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I的运算

- 1. 状态集I的 ϵ -闭包** 表示为 $\epsilon\text{-closure}(I)$ ，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 $\epsilon\text{-closure}(I)$
- 2. 状态集I的a弧转换** 表示为 $\text{move}(I,a)$ ，定义为状态集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

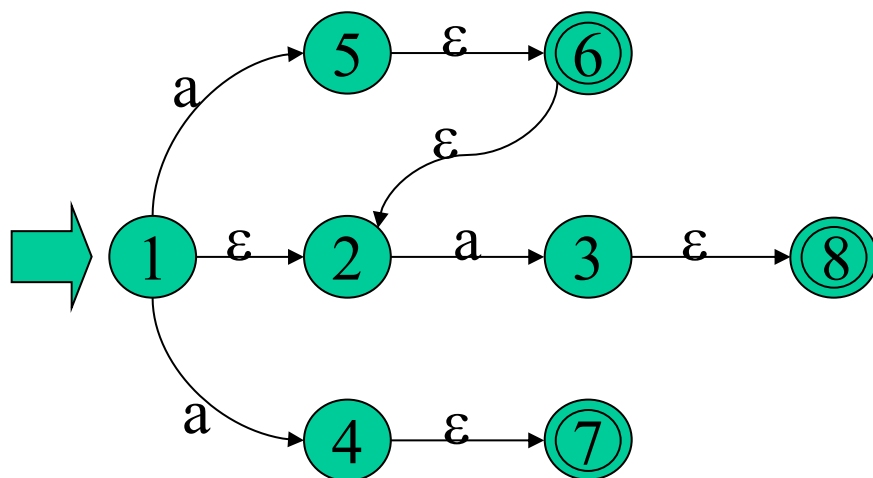
例子：对状态集合I的运算

$I = \{1\}$, $\varepsilon\text{-closure}(I) = \{1, 2\}$;

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

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

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



构造NFA N状态K的子集的算法

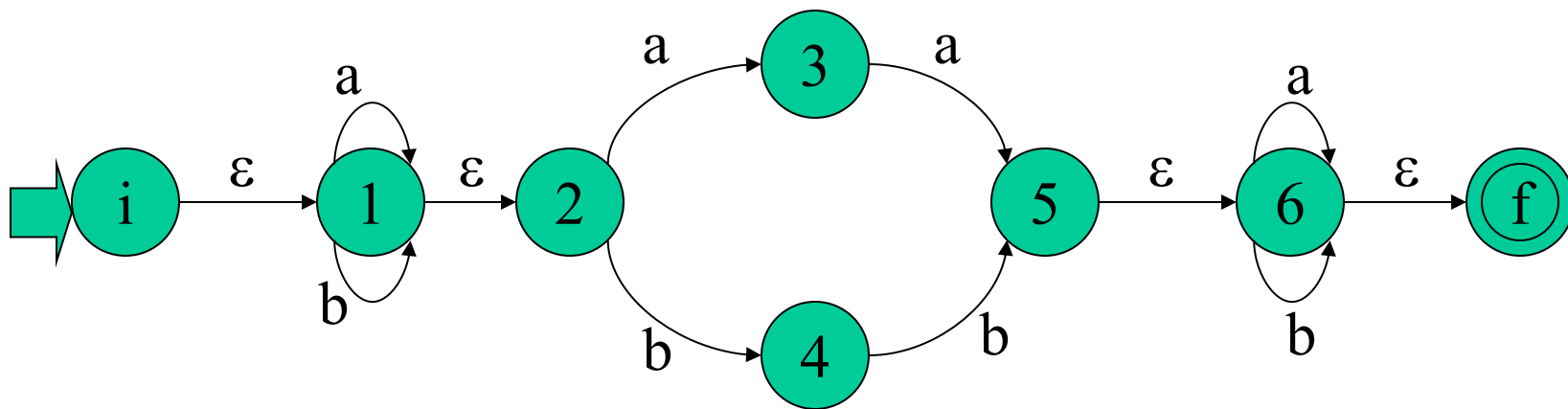
假定所构造的子集族为C，即 $C = (T_1, T_2, \dots, T_I)$ ，其中 T_1, T_2, \dots, T_I 为状态K的子集。

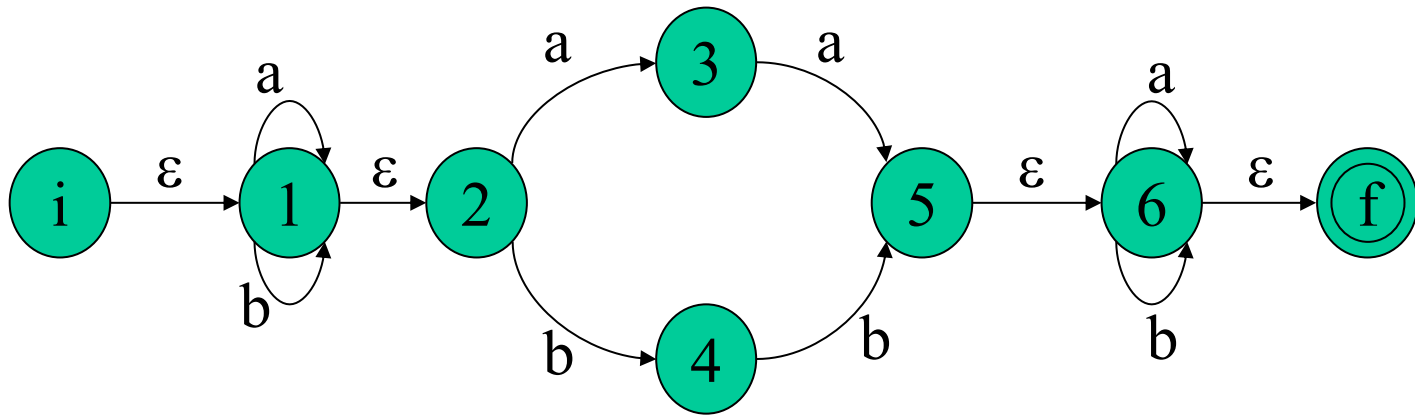
- 1 开始，令 $\varepsilon\text{-closure}(K_0)$ 为C中唯一成员，并且它是未被标记的。


```
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:=  $\varepsilon$ -closure(move(T,a));
        if U不在C中 then
            将U作为未标记的子集加在C中
    }
}
```

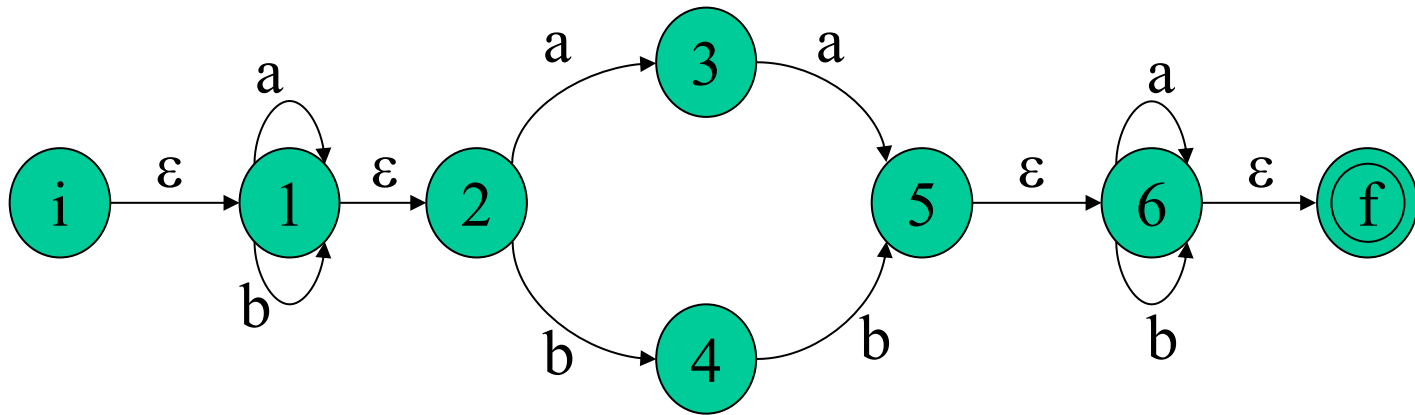
NFA的确定化

例子



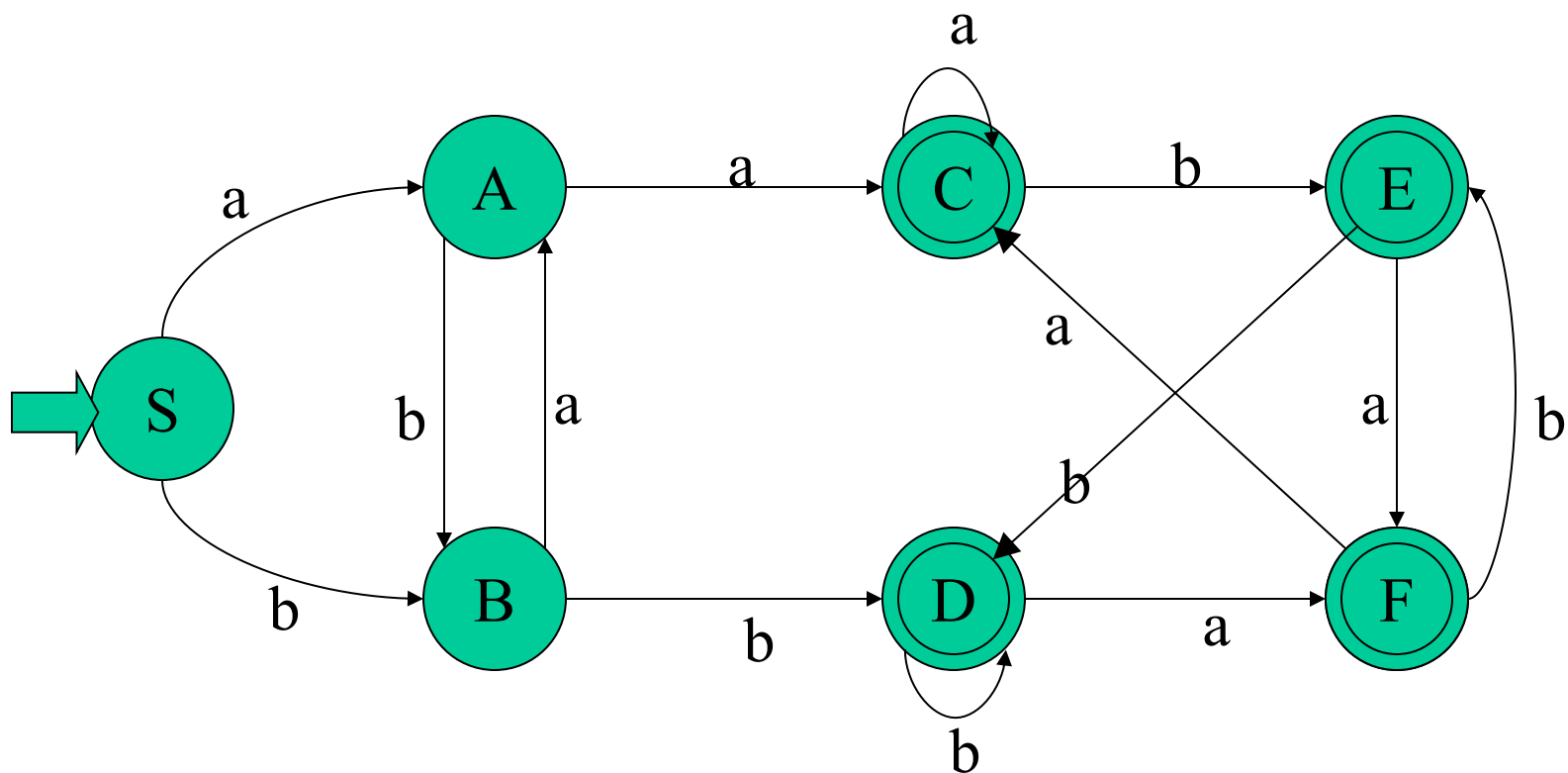


	Ia	Ib
{i,1,2}	{1,2,3}	{1,2,4}
{1,2,3}	{1,2,3,5,6,f}	{1,2,4}
{1,2,4}		
{1,2,3,5,6,f}		

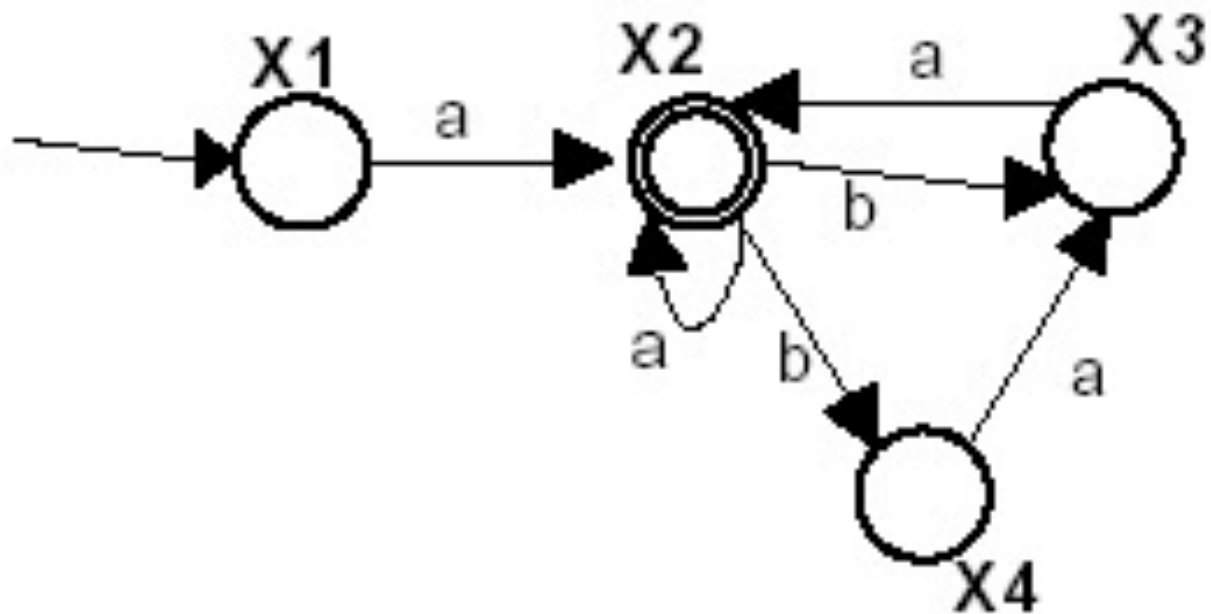


		Ia		Ib	
{i,1,2}	S	{1,2,3}	A	{1,2,4}	B
{1,2,3}	A	{1,2,3,5,6,f}	C	{1,2,4}	B
{1,2,4}	B	{1,2,3}	A	{1,2,4,5,6,f}	D
{1,2,3,5,6,f}	C	{1,2,3,5,6,f}	C	{1,2,4,6,f}	E
{1,2,4,5,6,f}	D	{1,2,3,6,f}	F	{1,2,4,5,6,f}	D
{1,2,4,6,f}	E	{1,2,3,6,f}	F	{1,2,4,5,6,f}	D
{1,2,3,6,f}	F	{1,2,3,5,6,f}	C	{1,2,4,6,f}	E

等价的DFA



NFA的确定化：例子



作业

- 1 (1)(3)