第3章 词法分析

Part II 自动机

主要内容

- DFA
- NFA
- NFA=>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)和

不确定的有穷自动机(Nondeterministic 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，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×Σ→K上的映射，即，如 f（ki，a）=kj，（ki∈K，kj∈K）就意味着，当前状态为ki，输入符为a时，将转换为下一个状态kj，我们把kj称作ki的一个后继状态；

4.S∈K是唯一的一个初态；

5.Z⊂ 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；

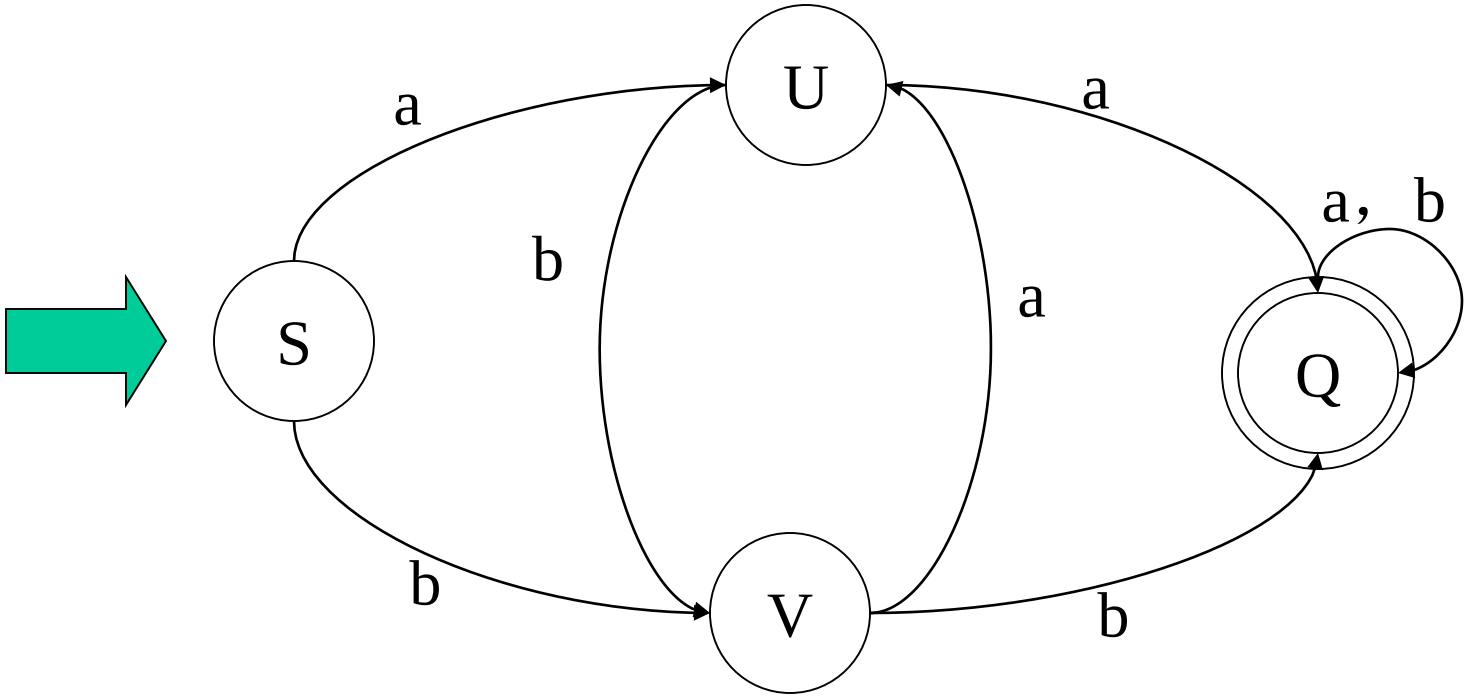
整个图含有唯一一个初态结点和若干个终态结点，初态结点冠以双箭头“=>”或标以“-”，终态结点用双圈表示或标以“+”；

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

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



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“=>”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
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

∑*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$T \in \sum$，$t_1 \in \sum^*$）在DFA M=（K，Σ，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：

$f(k_i, \varepsilon) = k_i$

∑*上的符号串t被DFA M接受：

M=（K，Σ，f，S，Z）

若t∈ ∑*，f（S，t）=P，其中S为M的开始状态，P ∈ 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。

f（S，baab）=f（f（S，b），aab）

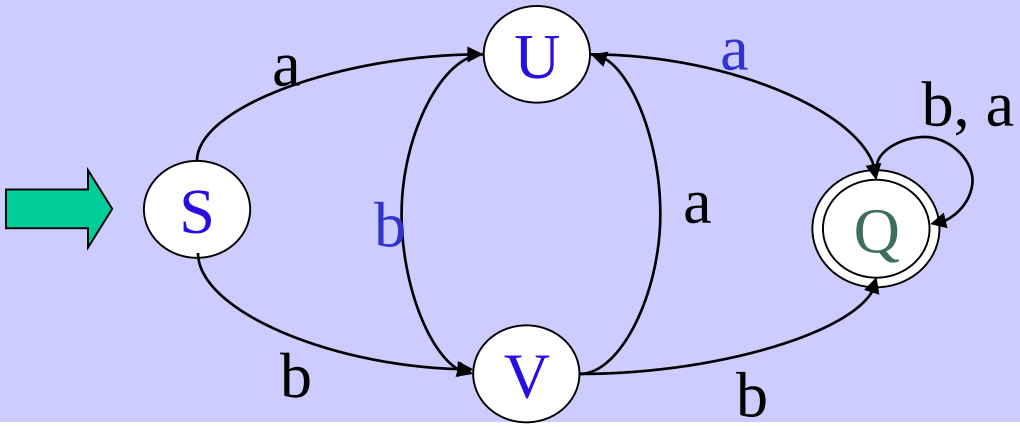
= f（V，aab）= f（f（V，a），ab）

=f（U，ab）=f（f（U，a），b）

=f（Q，b）=Q

Q属于终态。

得证。



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：

∑上一个符号串集V⊂∑*是正规的，当且仅当存在一个∑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，Σ，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’)
```

Review

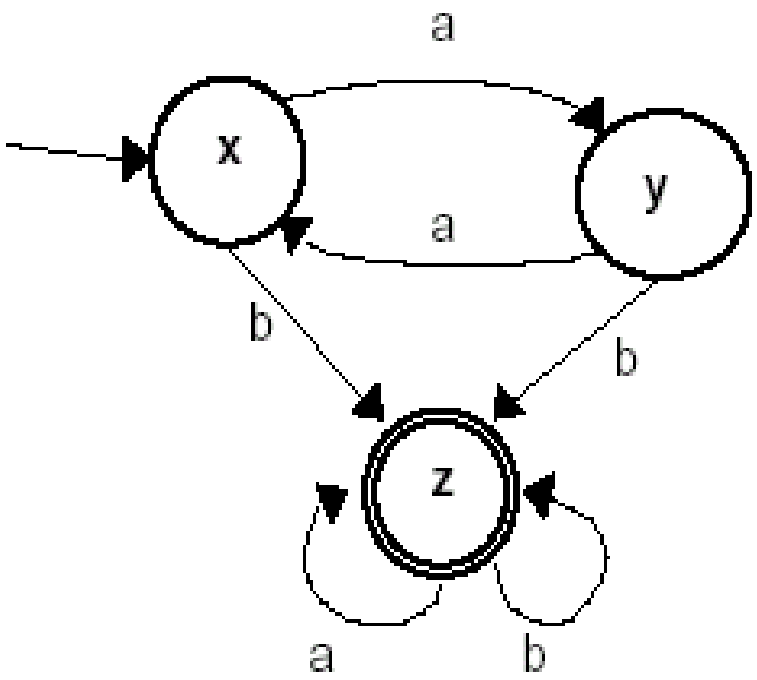
DFA M=（K，Σ，f，S，Z）

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.

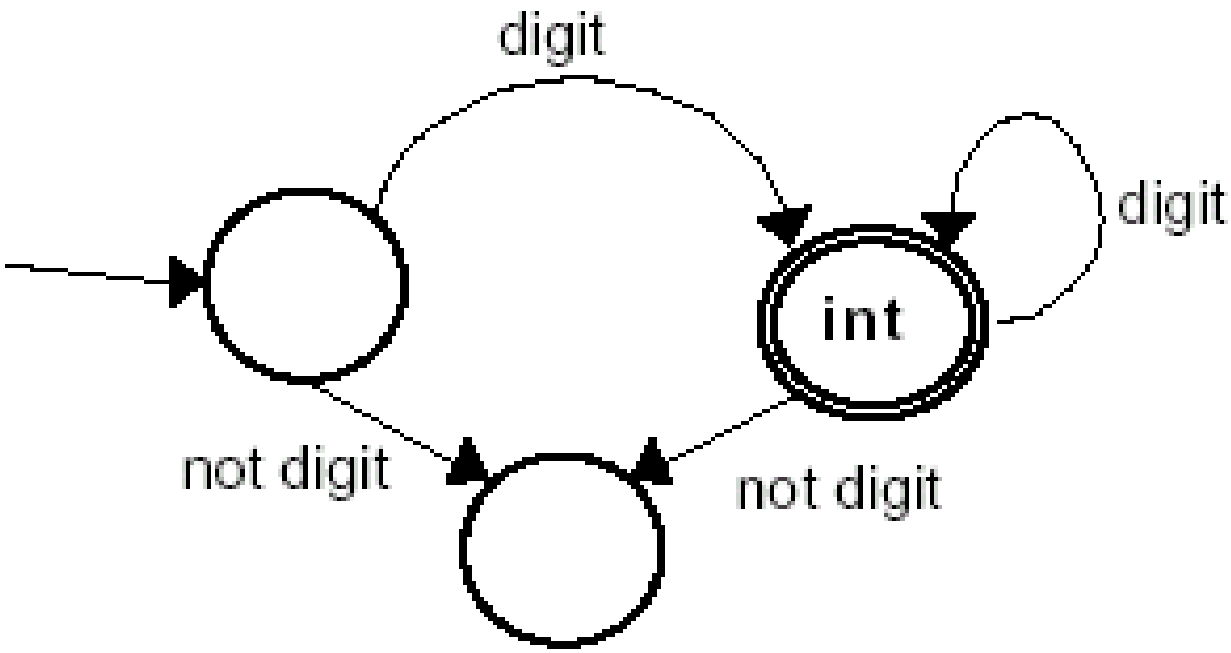
3) A finite set of transitions that specifies for each state and for each symbol of the input alphabet, which state to go to next.

DFA examples

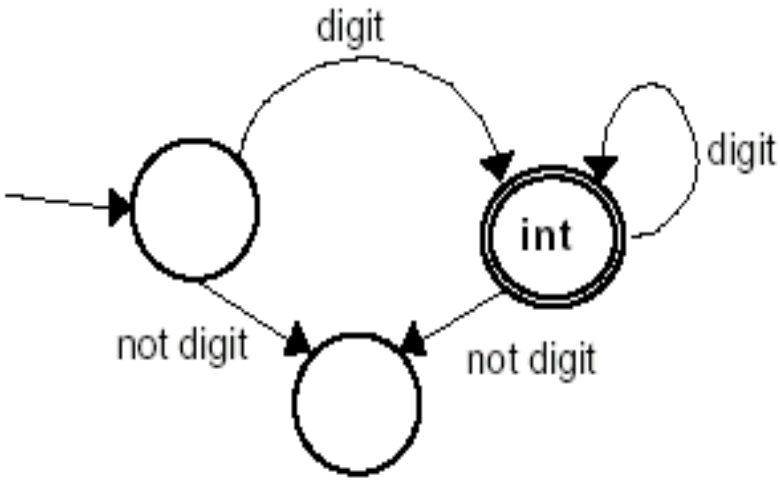


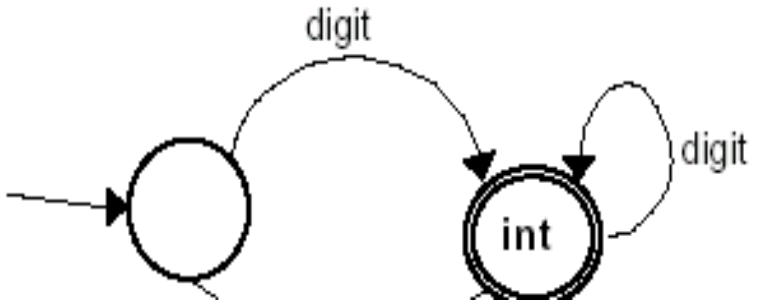
		a	b
(start)	x:	y	z
	y:	x	z
(final)	z:	z	z

DFA examples



FA 等价





不确定的有穷自动机NFA

定义：

NFA M=(K，∑，f，S，Z)，其中K为状态的有穷非空集，∑为有穷输入字母表，f为K×∑*到K的子集（2^K）的一种映射，S⊆K是初始状态集，Z ⊆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}

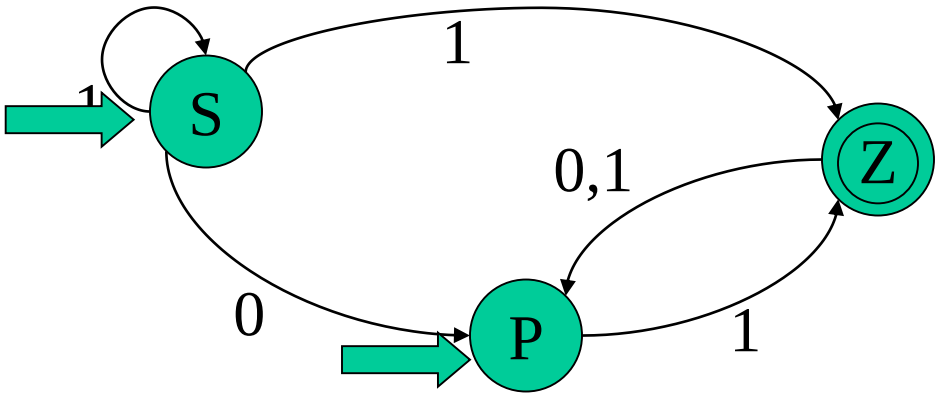
f（Z，0）={P}

f（P，1）={Z}

f（Z，1）={P}

f（S，1）={S，Z}

状态图表示

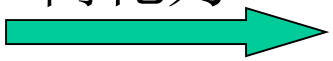


f（S，0）={P}
f（Z，0）={P}
f（P，1）={Z}
f（Z，1）={P}
f（S，1）
={S，Z}

矩阵表示

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

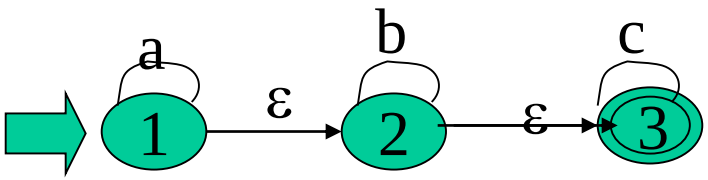
简化为



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

f为K× Σ* 到K的子集（2^K）的一种映射

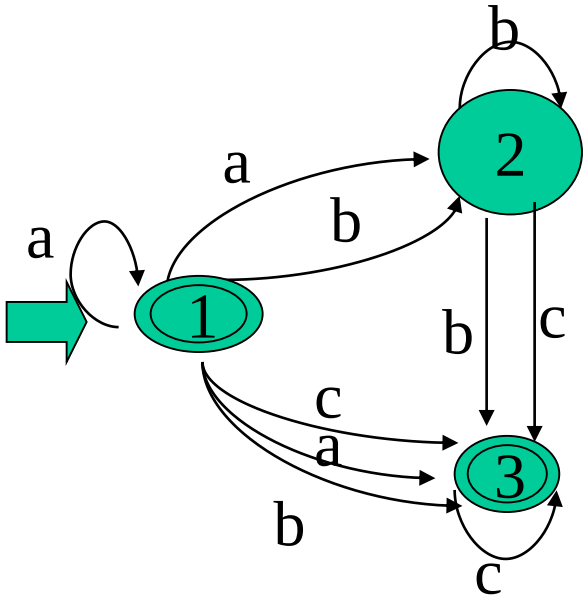
具有ε转移的不确定有穷自动机



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：



类似DFA, NFA M=(K，∑，f，S，Z)也有定义

∑*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$T \in \sum$，$t_1 \in \sum^*$）在NFA M上**运行**的定义为：

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

∑*上的符号串t被NFA M接受

若$t \in \sum^*$，$f(S_0, t)=P$，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结到某一终态结的道路，且这条道路上所有弧的标记字依序连接成的串(不理采那些标记为ε的弧)等于t，则称**t可为NFA M所识别**(读出或接受)。

若M的某些结既是初态结又是终态结，或者存在一条从某个初态结到某个终态结的道路，其上所有弧的标记均为ε，那么空字可为M所接受。

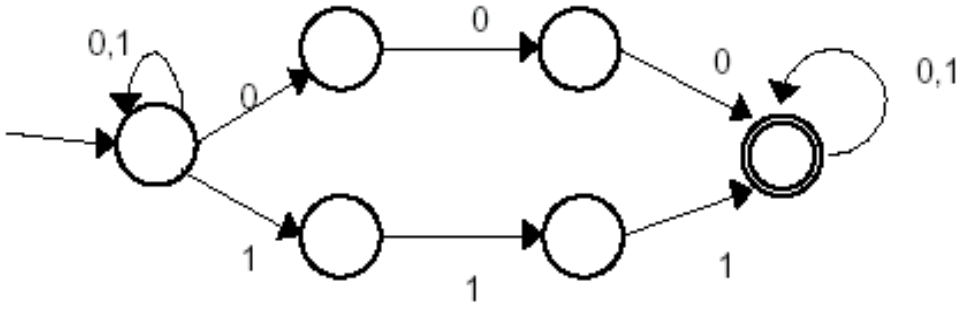
Examples

000

111

1010001

110000001



00

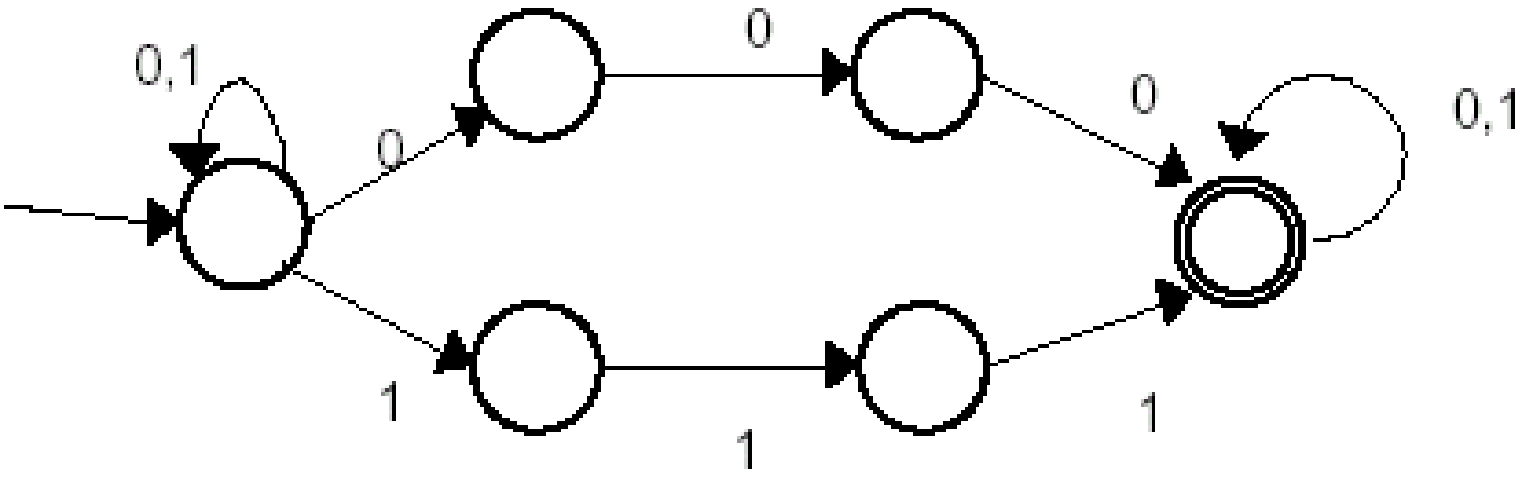
01100

NFA M所能接受的符号串的全体记为L(M)

结论：

∑上一个符号串集V⊂Σ^*是正规的，当且仅当存在一个∑上的不确定的有穷自动机M，使得V=L(M)

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



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

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，**NFA到相应的DFA的构造的基本思路是：*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, ∑, f, K_0, K_t), 按如下方法构造一个DFA M=(S, ∑, 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... S_j]表示S的元素，其中S_1, S_2, ... S_j是K的状态。并且约定，状态S_1, S_2, ... 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：

$d([S_1\ S_2\ ...\ S_j], a)= [R_1\ R_2\ ...\ R_t]$，其中

$\{R_1, R_2, ..., R_t\}= \varepsilon\text{-closure}(move(\{S_1,S_2,\ldots,S_j\},a))$

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

5. $S_t=\{[S_i\ S_k ... S_e]$，其中$[S_i\ S_k ... S_e]\in S$且

$\{S_i, S_k, \ldots, S_e\}\cap K_t \neq \Phi\}$

定义：对状态集合I的运算

1. 状态集合I的ε-闭包 表示为ε-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ε-closure(I)

2. 状态集合I的a弧转换 表示为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

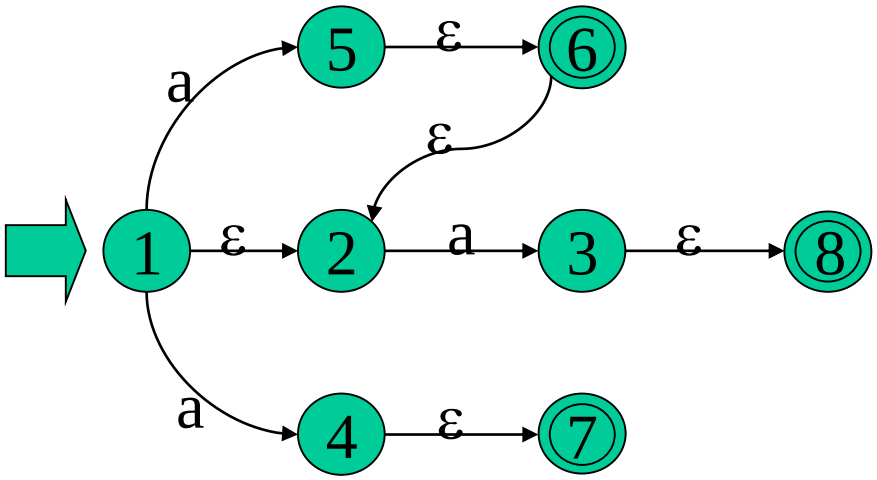
例子：对状态集合I的运算

I={1}, ε-closure(I)={1,2}；

I={5}, ε-closure(I)={5,6,2}；

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

ε-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,... T_I), 其中T_1, T_2,... T_I为状态K的子集。

1 开始，令ε-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:= ε-closure(move(T,a));

if U不在C中 ***then***

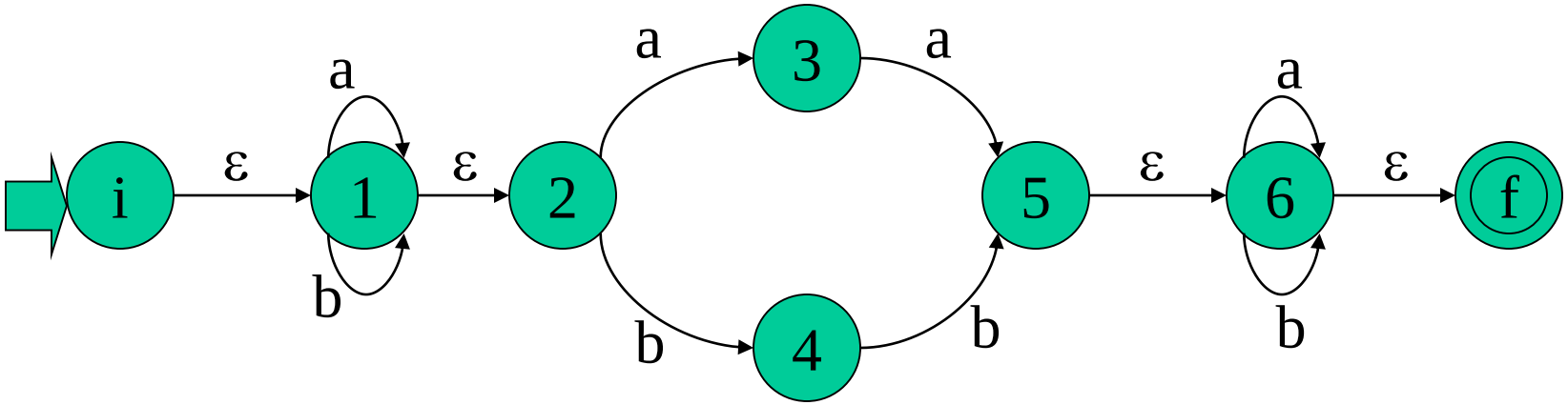
将U作为未标记的子集加在C中

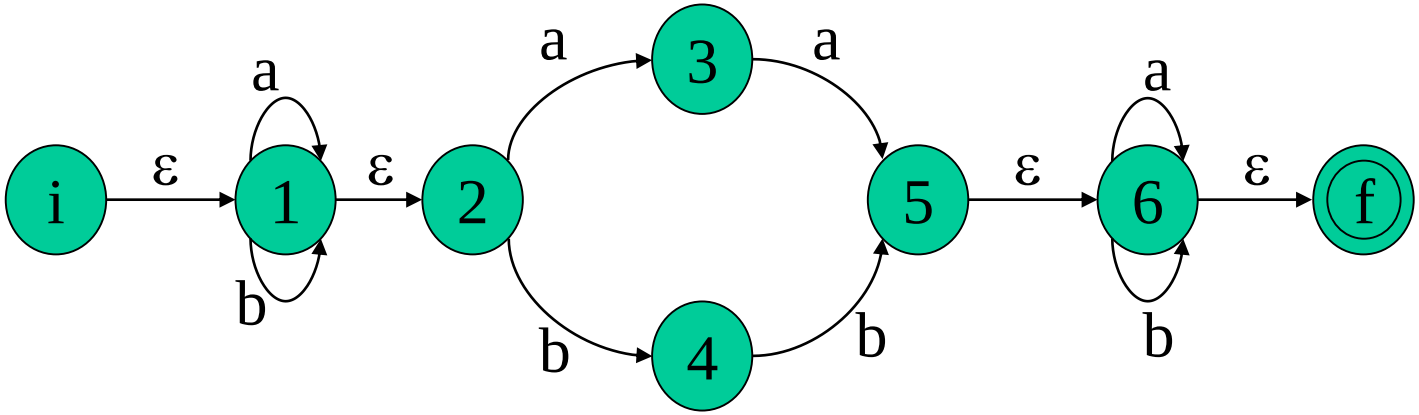
}

}

NFA的确定化

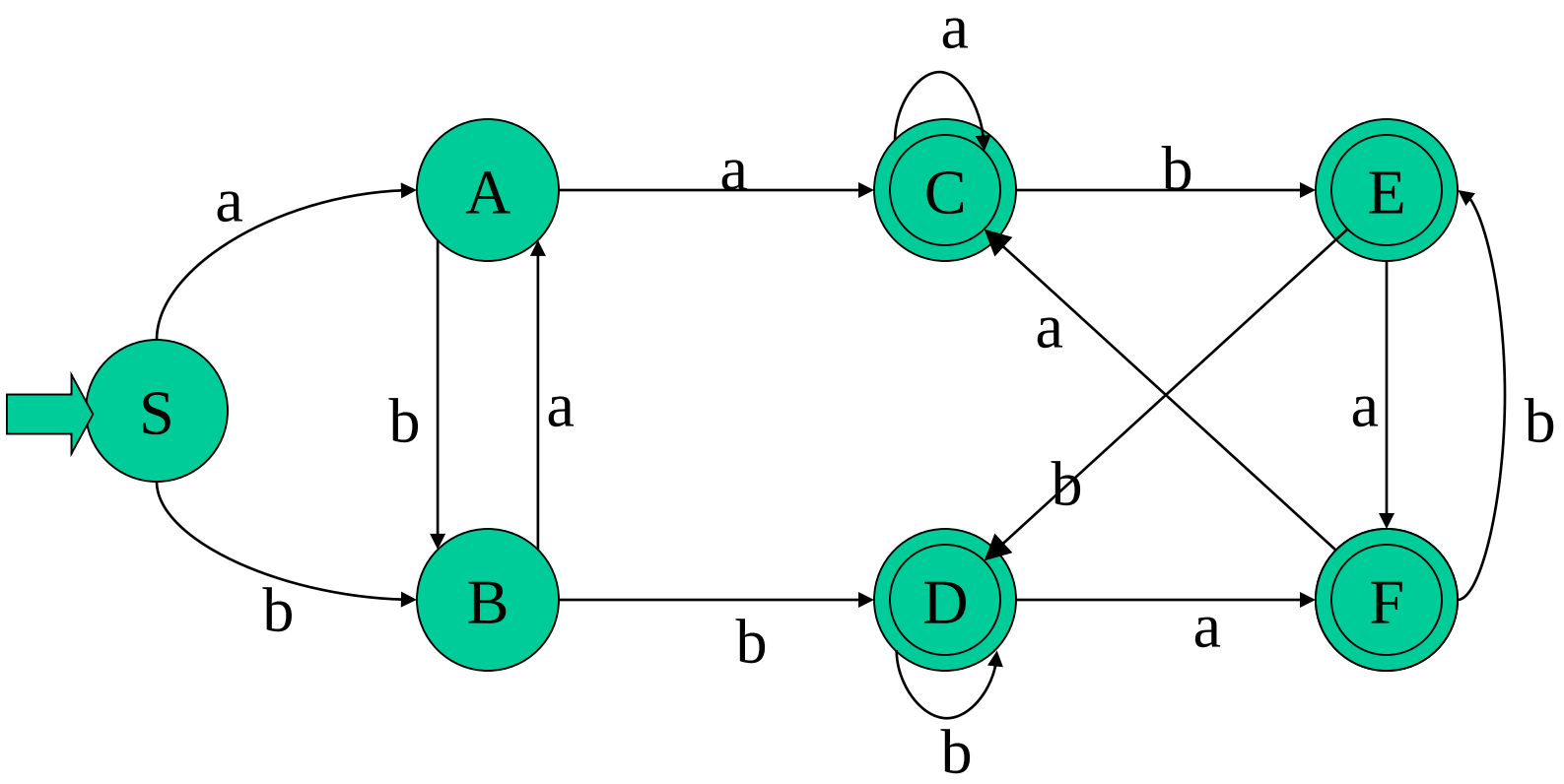
例子



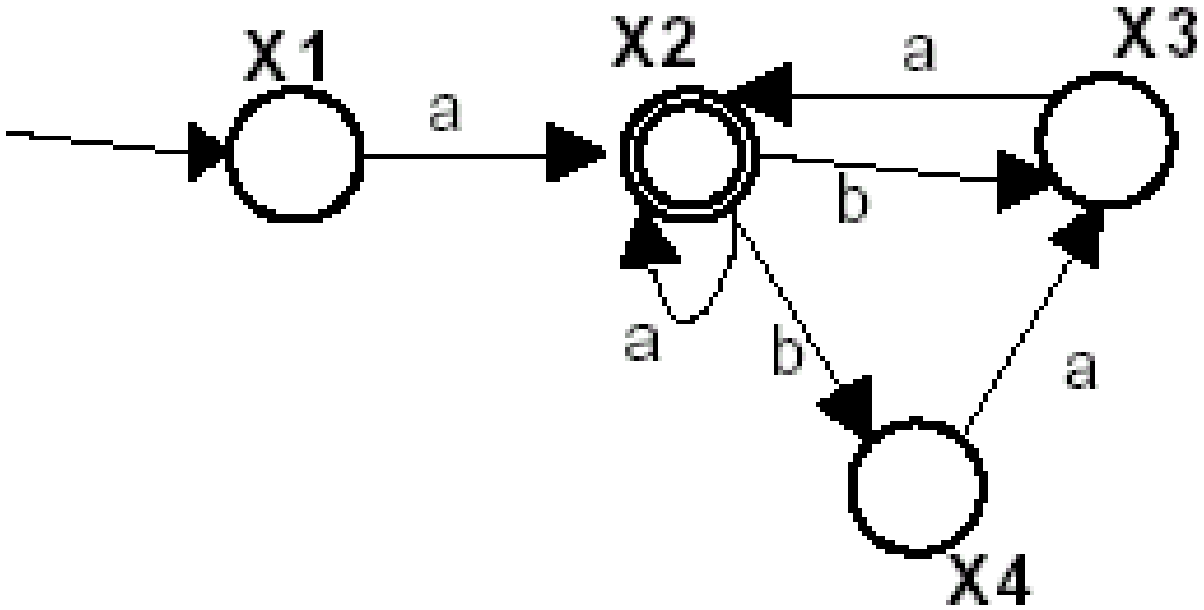


		Ia		Ib	
{i,1,2}	S	{1,2,3}	A	{1,2,4}	B
{1,2,3}	A	{1,2,3,5,6,f}	C	{1,2,4}	B
{1,2,4}	B	{1,2,3}	A	{1,2,4,5,6,f}	D
{1,2,3,5,6,f}	C	{1,2,3,5,6,f}	C	{1,2,4,6,f}	E
{1,2,4,5,6,f}	D	{1,2,3,6,f}	F	{1,2,4,5,6,f}	D
{1,2,4,6,f}	E	{1,2,3,6,f}	F	{1,2,4,5,6,f}	D
{1,2,3,6,f}	F	{1,2,3,5,6,f}	C	{1,2,4,6,f}	E

等价的DFA



NFA的确定化：例子



作业

- 1 (1)(3)