编译原理与技术

--词法分析II

刘爽

天津大学智算学部

Outline

- 词法分析器的作用
- 词法分析程序的设计与实现
  - 状态转换图
- 正则表达式和有限自动机
  - 正规/正则表达式（Regular Expression）
  - 有限自动机 （Finite Automata）
  - 正则文法（Regular Gramma）
- 词法分析程序的自动生成

问题的提出

- 将状态转换图的概念稍加形式化，更好的构造词法分析程序
- 有助于理解词法分析器的生成器，可以自动产生有效的词法分析器。（例如lex，flex，Jlex）

Review 语言上的运算定义

运算	定义和表示
L和M 的并	$L \cup M = \{s \mid s$ 属于L 或者s 属于 $M\}$
L和M 的连接	$LM = \{st \mid s$ 属于 L且　t 属于 $M\}$
L的 Kleene 闭包	$L^* = \cup_{i=0}^{\infty} L^i$
L的正闭包	$L^+ = \cup_{i=1}^{\infty} L^i$

(Kleene） closure $L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \ldots$

Positive closure $L^+ = L^1 \cup L^2 \cup L^3 \ldots$

$L^0 = \{ \varepsilon \}$

正则表达式与正则集 (Regular Set)

- 正则表达式可以详细说明单词符号的结构，可以精确地定义集合（正则集合）。正则表达式和正则集合的递归定义如下：
  1. ε 和Φ 都是Σ 上的正则式，其正则集为{ε} 和Φ；
  2. 任何**a**∈Σ，**a**是Σ的一个正则式，正则集为{**a**}；
  3. 假定**U** 和**V** 都是Σ 上的正则式，正则集为**L(U)** 和**L(V)**，那么，

     (U|V), (UV), (U)*也都是**S**上的正则式，正则集为**L(U) U L(V)**，**L(U)L(V)** 和**L(U)***；

  仅由有限次使用上述三步骤而定义的表达式才是Σ上的正则表达式，仅由这些正则表达式所表示的字集才是Σ上的正则集。

正则集合（正规集）：可以用一个正则表达式定义的语言。

正则表达式和正则集的例子

令Σ={a，b},

正则表达式	正则集合集
a	{a}
a|b	{a, b}
ab	{ab}
(a|b)(a|b)	{aa, ab, ba, bb}
a*	{ε ,a, aa, ……任意个a的串}
(a|b)*	{ε, a, b, aa, ab ……所有由a 和b组成的串}
(a|b)*(aa|bb)(a|b)*	{Σ*上所有含有两个相继的a或两个相继的b组成的串}

正则表达式和正则集—练习

- Σ={a,b} 则Σ上的正则表达式
  - $a^*ba^*ba^*ba^*$表示的正则集是?
  - $((\varepsilon|a)b^*)^*$表示的正则集是?

正则表达式的等价性

- 若两个正则表达式所表示的正则集合相同，则认为二者等价。两个等价的正则表达式U和V记为U=V。
  - 例如：　U= (a|b)，　V = (b|a)
  - 又如：　U= b(ab)* ，V =(ba)*b
  - 再如：　U= (a|b)* ，V =(a*|b*)*

正则表达式的性质

- 设U、V和W都是正则表达式，则如下关系普遍成立：
  - U|V = V|U　　（交换律）
  - U|(V|W) = (U|V)|W　　（结合律）
  - U(VW) = (UV)W　　（结合律）
  - U(V|W) = UV | UW　　（分配律）
    (V|W)U = VU |WU;
  - εU = Uε=U
  - r|r=r　　r*=ε|r|rr|…　“或”的抽取律

- 运算优先级和结合性:
  - *　高于 "连接" 高于 |
  - |　具有交换律、结合律
  - “连接”　具有结合律、分配律
  - ( )　　指定优先关系

有限自动机

有限自动机

- 有限自动机（也称有穷自动机）作为一种识别装置，它能准确地识别正则集合，即识别正则文法所定义的语言与正则表达式所表示的集合，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。
- 有限自动机分为两类：
  - 确定有限自动机 Deterministic Finite Automata（DFA）
  - 不确定有限自动机 Nondeterministic Finite Automata（NFA）
- 其中“不确定”的含义是：对于某个输入符号，在同一状态上存在不止一种转换。
- 确定和不确定有限自动机都能而且仅能识别正则集合，即它们能够识别正则表达式所表示的语言。

确定有限自动机

确定有限自动机 (DFA)

- 一个确定的有限自动机(DFA)是一个五元式： $M=(S, \sum, \delta, s_0, F)$ 其中
  - S是一个有限集，它的每个元素称为一个状态。
  - $\sum$是一个有穷字母表，它的每个元素称为一个输入字符
  - $\delta: S\times\sum \rightarrow S$ 是一个单值映射 （确定性）。
    - $\delta(s, a)=s'$意味着： 当现行状态为s、输入字符为a时， 将转换到下一个状态s'。我们称s'为s的一个后继状态。
  - $s_0 \in S$， 是唯一的初态。
  - $F \subseteq S$， 是一个终态集（可空）

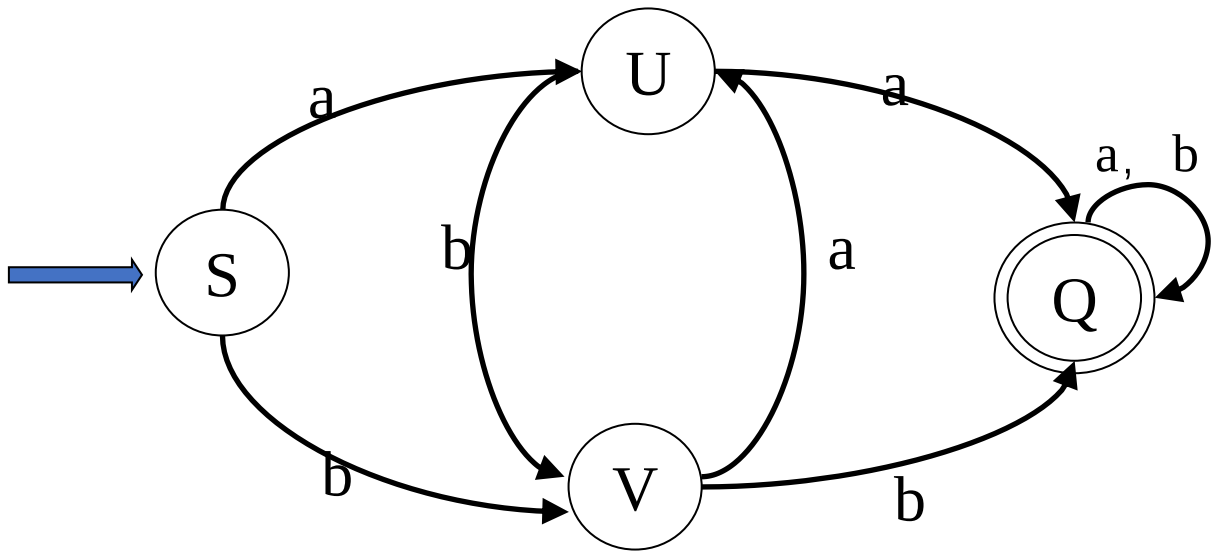
确定的有限自动机—表示方式

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

δ（S，a）=U　　δ（V，a）=U
δ（S，b）=V　　δ（V，b）=Q
δ（U，a）=Q　　δ（Q，a）=Q
δ（U，b）=V　　δ（Q，b）=Q

状态 \ 字符	a	b
S	U	V
U	Q	V
V	U	Q
Q	Q	Q

图（1）状态转换矩阵



图（2）状态转换图

δ(s,a)值------状态转换矩阵值。如图（1）所示：行--状态，列--输入字符

一个DFA也可以表示为一张（确定的）状态转换图，如图（2）所示

练习

DFA M 可用一张（确定的）状态转换图表示。

- DFA M=({A,B,C,D},{a,b}, δ,A,{D})

$$\delta(A,a)=B \quad \delta(A,b)=A$$
$$\delta(B,a)=B \quad \delta(B,b)=C$$
$$\delta(C,a)=B \quad \delta(C,b)=D$$
$$\delta(D,a)=B \quad \delta(D,b)=A$$

画出其对应的状态转换图和状态转换矩阵

确定有限自动机（DFA）接受的字符串

- DFA接受的字符串
  - 对于∑*中的任何字符串t，若存在一条从初态到某一终态的路径，且这条路径上所有弧的标记字符连接成的字符串等于t, (即若$t=t_1t_2...t_n \in \sum^*$，$\delta(S, t_1)=P_1$，$\delta(P_1, t_2)=P_2$，…, $\delta(P_{n-1}, t_n)=P_n$，其中S为**M**的开始状态, $P_n \in Z$, Z为终态集）则称t可被**DFA M接受(识别)**
  - 若M的初态同时又是终态，则空字符串可为M所识别。
  - 一个**DFA M** 所能识别的所有的字符串的集合记为：**L(M)**
  - Σ上的一个字集**V ⊆ Σ*** 是正则的，当且仅当存在Σ上的**DFA M ,**使得**V=L(M)**
- ∑*上的符号串t，(我们将它表示成t_1t_x的形式，其中$t_1 \in \sum$，$t_x \in \sum^*$) 在**DFA M** 上运行的定义为：
  - $\delta(Q, t_1t_x)= \delta(\delta(Q, t_1), t_x)$ 其中Q∈S （是**M**的开始状态）

确定有限自动机—例子

- 例：证明t=baab被例中的DFA所接受。

δ (S,baab)

= δ (δ (S,b),aab)

= δ (V,aab)

= δ (δ (V,a),ab)

= δ (U,ab)

= δ (f(U,a),b)

= δ (Q,b)

=Q

Q属于终态, 找到了从M的初态到终态的一条路径，该路径上的字符连接成的字符串为t。

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

δ（S，a）=U　　δ（V，a）=U
δ（S，b）=V　　δ（V，b）=Q
δ（U，a）=Q　　δ（Q，a）=Q
δ（U，b）=V　　δ（Q，b）=Q

非确定有限自动机

非确定有限自动机 （NFA）

- 一个非确定有限自动机(NFA)是一个五元式： $M=(S, \sum, \delta, S_0, F)$ 其中
  - S是一个有限集，它的每个元素称为一个状态。
  - $\sum$是一个有穷字母表，它的每个元素称为一个输入字符
  - $\delta: S\times\sum^* \rightarrow 2^S$ 是一个从$S\times\sum^*$至**S子集**的单值映射。
  - $S_0 \subseteq S$，是一个非空的**初态集**
  - $F \subseteq S$，是一个终态集（可空）

若δ是一个多值函数，且输入可允许为ε，则有穷自动机是不确定的，即在某个状态下，对于某个输入字符存在多个后继状态。

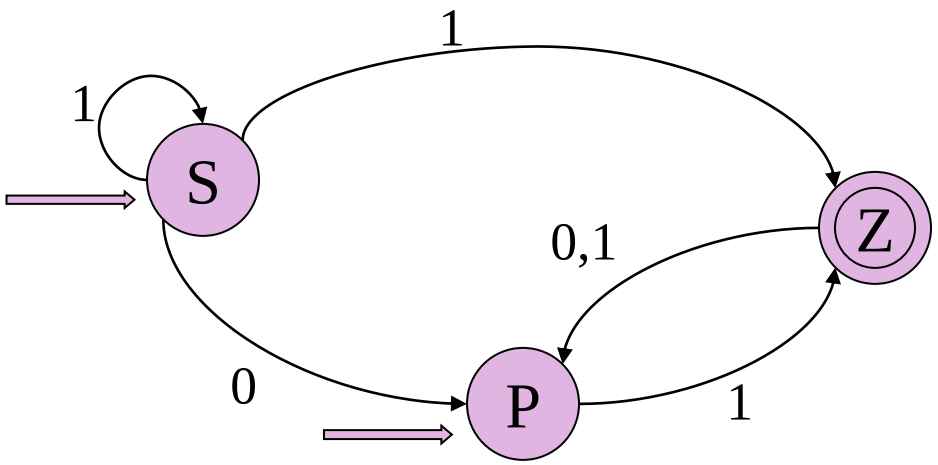
非确定的有穷自动机—表示方式

NFA N=({S,P,Z},{0,1},δ,{S,P},{Z})

- δ(S,0)={P}
- δ(S,1)={S, Z}
- δ(P,1)={Z}
- δ(Z,0)={P}
- δ(Z,1)={P}

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

图（1）状态转换矩阵



图（2）状态转换图

δ (s , a) 值------状态转换矩阵值。如图（1）所示：行--状态，列--输入字符

一个NFA也可以表示为一张（确定的）状态转换图，如图（2）所示

NFA--练习

画出下列NFA的状态转换矩阵和状态转换图

NFA M=({S,Q,U,V,Z},{0,1}, δ,{S},{Z});

δ(S,0)={V,Q}　　**δ(S,1)={U,Q}**

δ(U,0)=Φ　　**δ(U,1)={Z}**

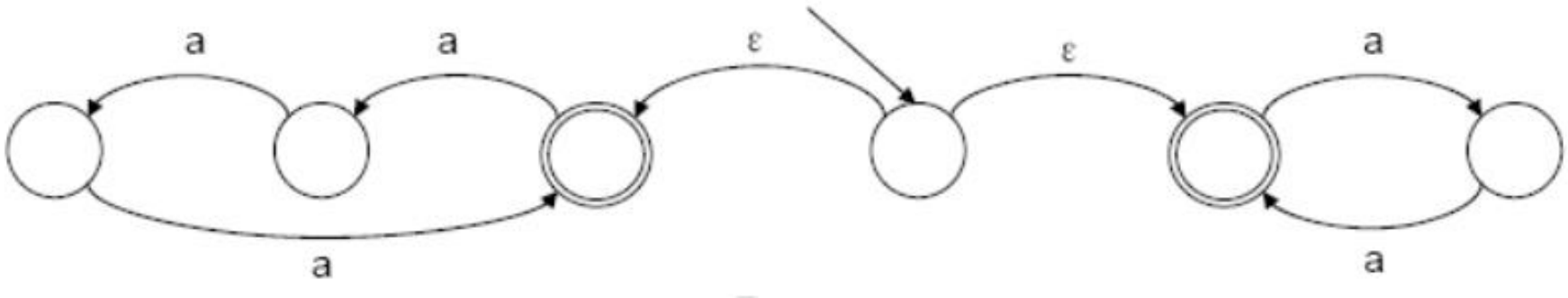
δ(V,0)={Z}　　**δ(V,1)= Φ**

δ(Q,0)={V,Q}　　**δ(Q,1)={U,Q}**

δ(Z,0)={Z}　　**δ(Z,1)={Z}**

非确定有限自动机(NFA)接受的字符串

- 对于∑*中任何字α，若存在一条从某一初态到某一个终态的路径，且这条路径上所有弧的标记字符依序连接成的字符串等于α，则称α可为NFA M所识别（接受）。
- 若M的某些结点既是初态又是终态，或存在一条从某一初态到某一终态的ε路径，那么空字ε可为M所识别。

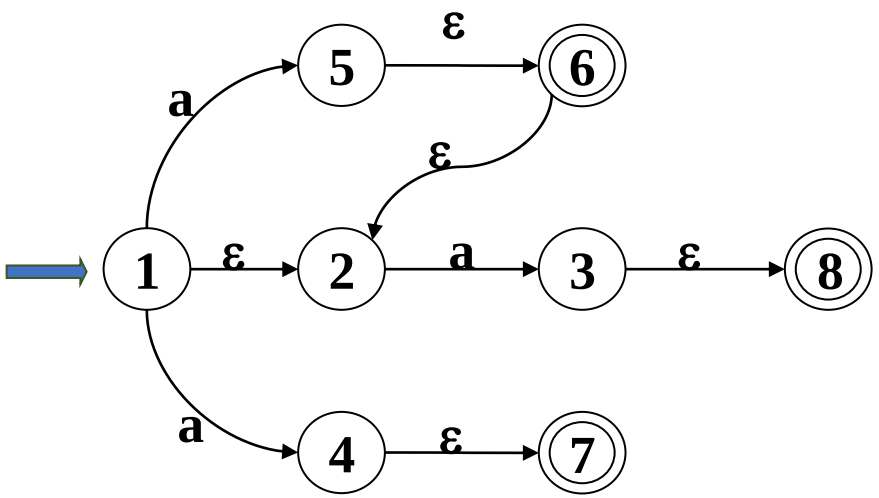


NFA的确定化

NFA的确定化

- 显然，DFA是NFA的特例，但是对于每个NFA M都存在一个DFA M''，使L(M)=L(M'')。
- 定义对状态集合I的几个有关运算：
  - 状态集合I的**ε-闭包**，表示为**ε-closure(I)**，定义为一状态集，包括
    (1)状态集合I的任何状态s都属于**ε-closure(I)**；
    (2)状态集合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**a弧转换** I_a **= ε-closure(move(I, a)) = ε-closure(J)**，其中**move(I, a)** 为状态集合J，其定义是所有那些可从I的某一状态经过一条a弧而到达的状态的全体。

ε-闭包--例子



- I={1}, ε-closure(I)={1,2};
- I={5}, ε-closure(I)={5,6,2};
- I={1,2},
  move({1,2},a)={5,3,4}=J;
  I_a=ε-closure({J})=ε-closure({5,3,4})={2,3,4,5,6,7,8};

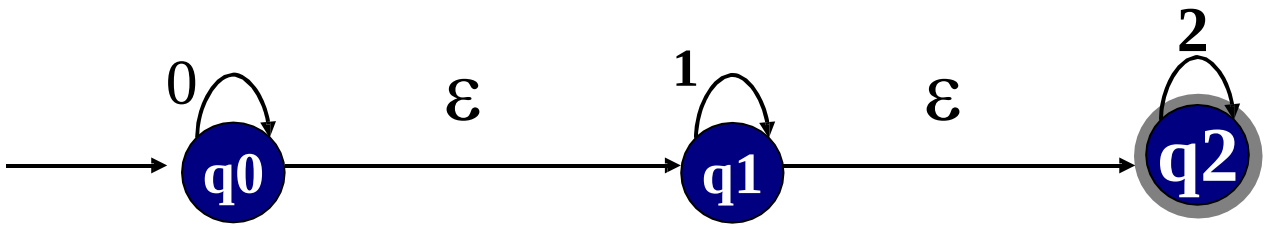
具有ε–转移的NFA和不具有ε–转移的NFA的等价性

定理1：对任何一个具有ε–转移的NFA M，一定存在一个不具有ε–转移的 NFA M’, 使

L(M’)=L(M)。

从M出发构造一个不具有 ε–转移的 NFA M’，使得L(M’)=L(M)。

- 例： 设NFA M=(Q, Σ, f, {q0}, F), 其中 Q={q0,q1,q2}， F={q2}



解

解：令M’=（Q, Σ, f’, q0, F’），其中Σ, Q, q0 的意义同M中完全 一样。

1)F’包含M的终态集F，其次若M中从q 出发有一条到达某终态的ε路径， 则将q0加在F’中。

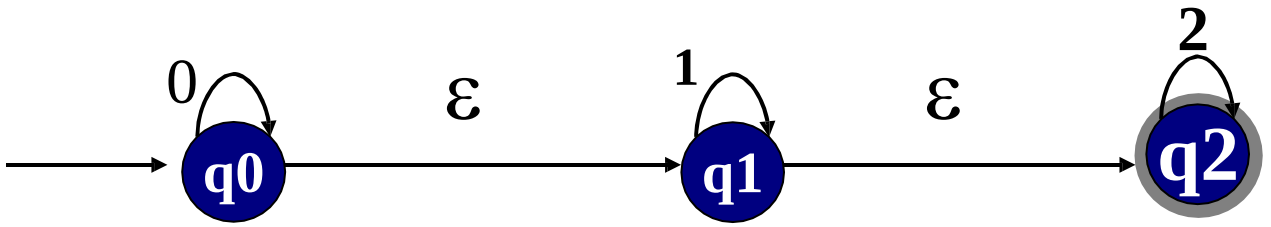
$$F' = \begin{cases} F \cup \{q\}, & \text{若} \varepsilon\text{–closure}(q)\text{包含}F\text{的一个状态} \\ F, & \text{否则。} \end{cases}$$

终态集　F’ = {q0,q1,q2},

ε-closure(q0)={q0,q1,q2}

ε -closure(q1)={q1,q2}

解



2) f’(q,a)={q’ | q’为从q出发先经若干ε箭弧，接着经一个标记为a的箭弧，再经若干ε箭弧组成的道路所能到达的状态。}

f’(q0 ,0) = {q0,q1,q2}

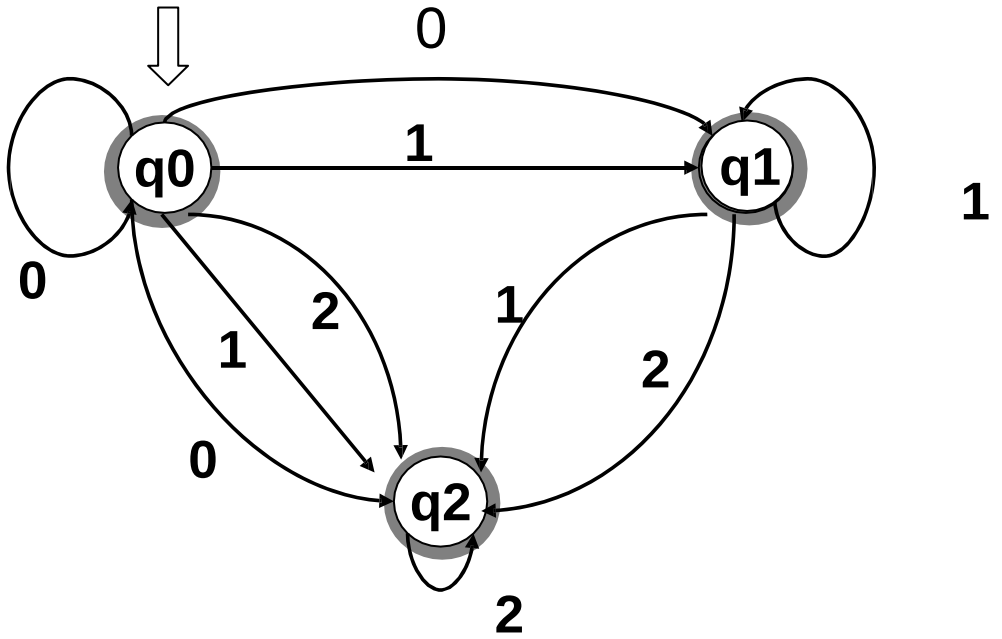
f’(q0 ,1) = {q1,q2}

f’(q0 ,2) = {q2}

f’(q1 ,1) = {q1,q2}

f’(q1 ,2) = {q2}

f’(q2 ,2) = {q2}



状态图

不具有ε–转移的NFA转具有ε–转移的NFA

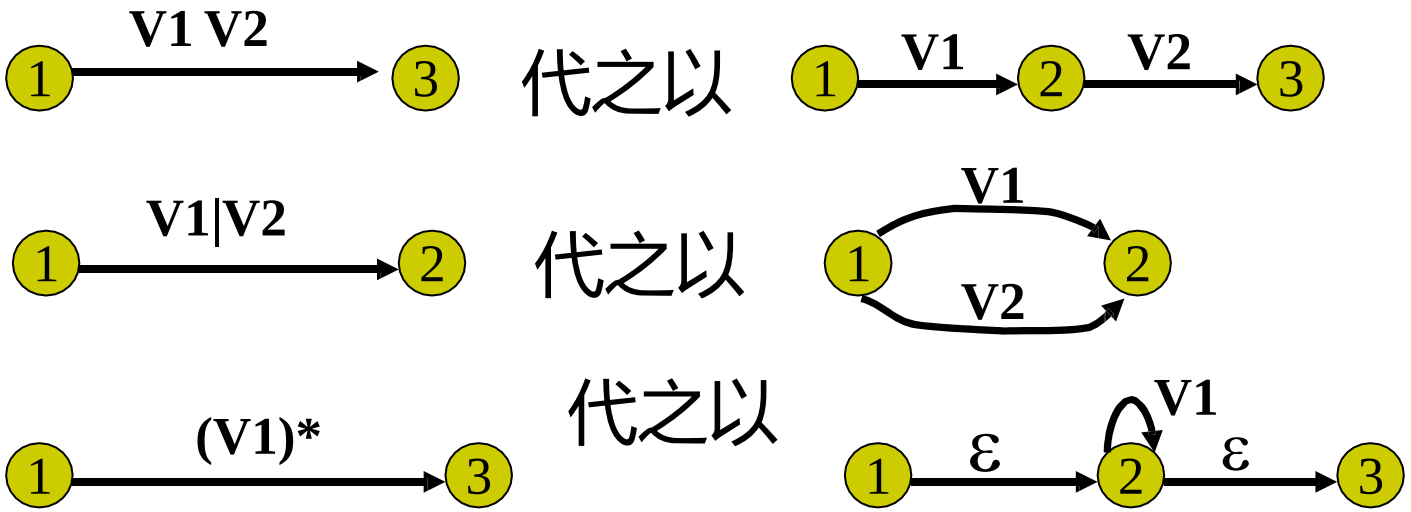
- 定理1：对任何一个不具有ε–转移的NFA M'=(Q, Σ, f', q0, F')，一定存在一个具有ε–转移 的 NFA M, 使
  - L(M) = L(M')

从M'出发构造一个具有ε–转移的 NFA M，使得L(M) = L(M')

- 步骤：
  （1）引进初态X和终态Y，X, Y ∉ S, 从X到S_0中任意状态连接一条ε箭弧，从F中任意状态连接一条ε箭弧到Y。

不具有ε–转移的NFA转具有ε–转移的NFA

（2）对M的状态转换图进一步实施如下替换，其中状态2是新引入状态。重复该过程，直到每条箭弧上标记ε，或者为∑中的单个字符。



将 ε–NFA转化为等价的 DFA

- 定理2：对于字母表Σ上任何一个具有ε–转移的NFA M, 一定存在一个的DFA M’, 使得：L(M’) = L(M)。

- NFA M =(S,Σ,f,S0,Z’) 用构造ε–closure(T) 的方法构造DFA M’ = （S’,Σ,f’,q0,Z’）：

- 基本思想：

1) 首先从S0出发，仅经过任意条ε箭弧所能到达的状态所组成的集合I作为M’的初态q0.

2) 分别把从q0（对应于M的状态子集I）出发，经过任意a∈Σ 的a弧转换I_a 所组成的集合作为M’ 的状态，如此继续，直到不再有新的状态为止。

将 ε–NFA转化为等价的 DFA (cont)

- **I_a = ε-closure(move(I,a)) = ε-closure(J)**

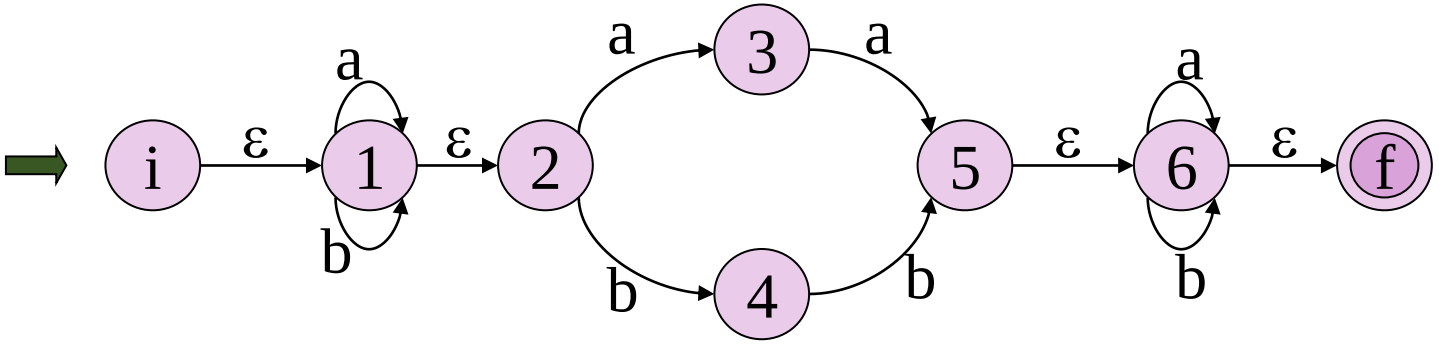
- 假定∑有k个字母。我们构造一张表，有k+1列。
  - 表的每行首列为**ε-closure(X)**，$X \subset \sum$，且未曾出现在表的第一列。
  - 表的第i+1列为**I_{ai}**, a_i 为∑中的第i个字符。
  - 重复上述过程填表，因为∑子集有限，所以该过程一定会终止。

DFA和NFA的等价性

- 定理3：对任何一个**NFA M** =(S, Σ, f, S_0, F)，都存在一个**DFA M'** =(S', Σ, f', s_0, F')，使得**L(M')=L(M)**。
  - 证明的思想是由**M**出发构造等价的**M'**，办法是让**M'** 的一个状态对应于**M**的一个状态集合。
  - 即若 δ**(s, a)={s1, s2, …, sk},**
  - 我们将集合**{s1, s2, …, sk}**作为一个整体看作**M'**中的一个状态，即**S'**中的一个元素。

NFA确定化的例子

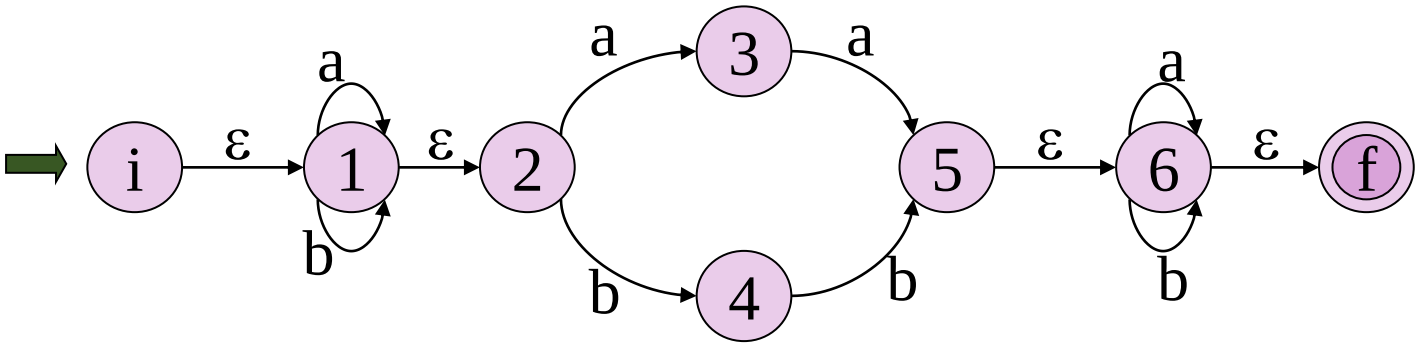
I_a = ε-closure(move(I,a)) = ε-closure(J)



I	Move(I, a)	I_a	Move(I, b)	I_b

NFA确定化的例子

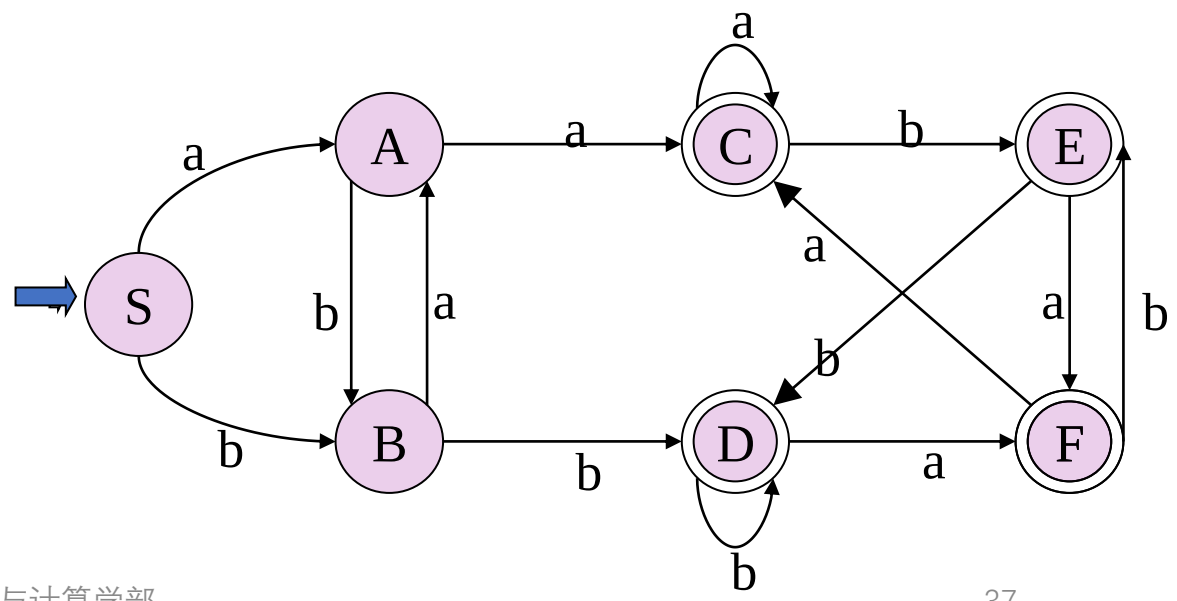
I_a = ε-closure(move(I,a)) = ε-closure(J)



包含原初始状态i的状态子集为DFA的初态

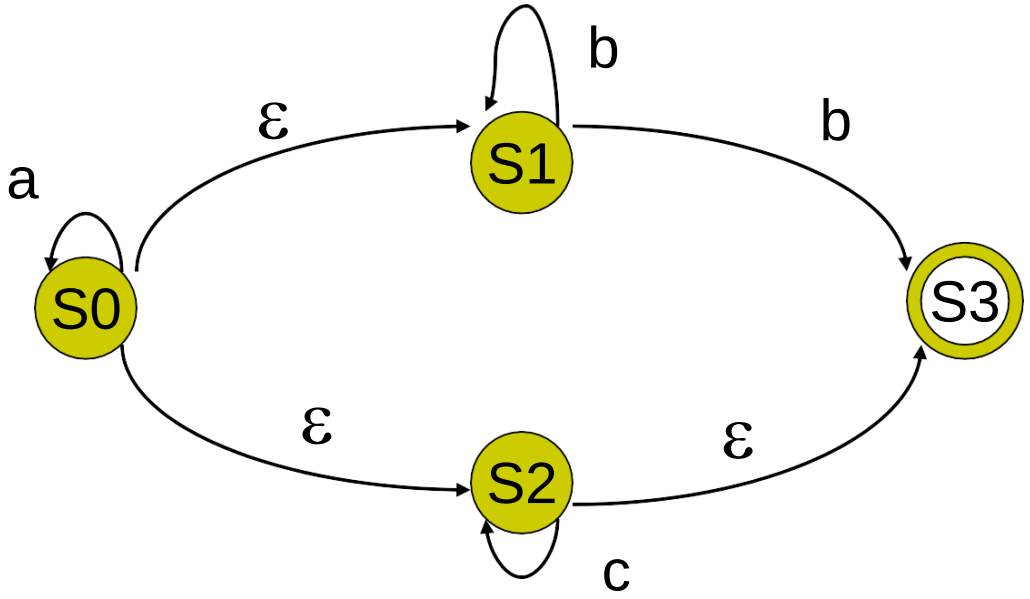
包含原终止状态f的状态子集为DFA的终态

	I_a	I_b
{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



练习

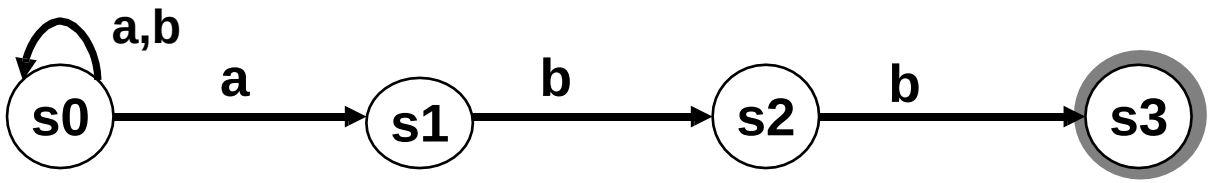
- 将下列ε– NFA确定化:



练习

NFA M=({S0,S1,S2,S3},{a,b},f,S0,{S3})

将M转换成其等价的DFA



DFA最小化

DFA的最小化

- DFA的最小化(DFA的化简): 一个有穷自动机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。
  - 寻找一个状态数比M少的 DFA M’，使得L(M) = L(M’)
- 最小状态DFA
  - 不存在多余状态(死状态)
  - 不存在互相等价（不可区别）的两个状态
- 两个状态s和t等价：从s出发可以读出某个字符串w而到达终态，从t出发也可以读出字符串w而到达终态；反之亦然。
- 两个状态s和t可区别：不满足
  - 兼容性（一致性）——同是终态或同是非终态
  - 传播性（蔓延性）——从状态s出发读入某个a ($a \in \Sigma$)和从状态t出发读入某个a到达的状态等价。

DFA的最小化—算法

- 算法的核心
  - 把M的状态集K分成不相交的子集。
- 结论
  - 接受L的最小状态有穷自动机不计同构是唯一的。
- DFA M = （S, ∑, δ, s_0, S_t), 最小状态DFA M’

  1.构造状态的初始划分∏：终态S_t 和非终态S- S_t两组

  2.对∏施用传播性原则构造新划分$\prod_{new}$

  3. 如$\prod_{new}$ == ∏,则令 $\prod_{final}$= ∏ 并继续步骤4，否则∏ := $\prod_{new}$重复2

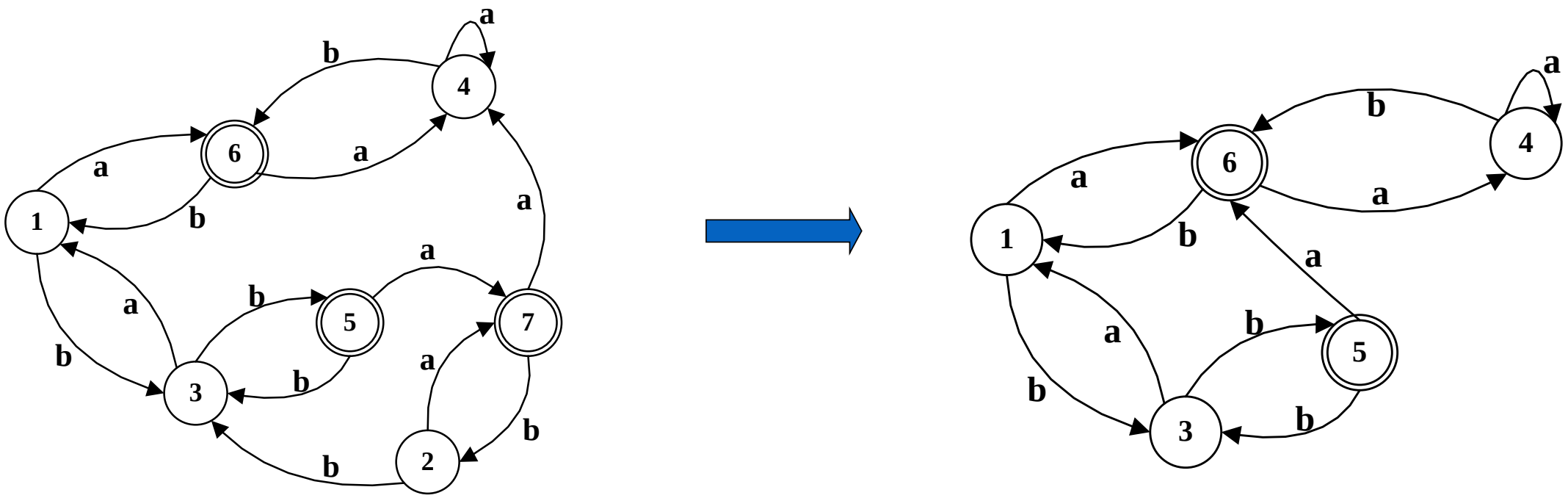
  4.为$\prod_{final}$中的每一组选一代表，这些代表构成M’的状态。若s是一代表,且δ(s,a)=t, 令r是t组的代表，则M’中有一转换δ’(s, a)=r。 M’ 的开始状态是含有s_0的那组的代表，M’的终态是含有s_t的那组的代表

  5.去掉M’中的死状态

DFA的最小化—算法（cont）

- 对Π施用传播性原则构造新划分Π_{new}步骤：
  - 假设Π被分成m个子集$\Pi=\{S_1, S_2, ..., S_m\}$, 且属于不同子集的状态是可区别的，检查$\Pi$的每一个子集$S_i$, 看是否能够进一步划分。
  - 对于某个S_i, 令Si={s1, s2, ..., sk}, 若存在数据字符a使得I_a不全包含在现行Π的某一子集S_j中，则将S_i一分为二：即假定状态s1, s2，经过a弧分别达到状态t1, t2, 且t1, t2属于现行Π的两个不同子集，那么将S_i分成两半，一半含有s1: $S_{i1}=\{s \mid s \in S_i$ 且s经a弧到达t1所在子集中的某状态$\}$;
    另一半含有s2: $S_{i2}=S_i-S_{i1}$
  - 由于t1, t2是可区别的，即存在w, 使得t1读出w而停于终态，t2读不出w或读出w却未停于终态。因而aw可以将s1, s2区分开。也就是说S_{i1}和S_{i2}中的状态时可区别的。
  - $\Pi_{new}=\{S_1, S_2, ... S_{i1}, S_{i2}, ..., S_m\}$

DFA的最小化—例子



(1) 初始划分：一个由终态组成，一个由非终态组成 P0=({1,2,3,4},{5,6,7})

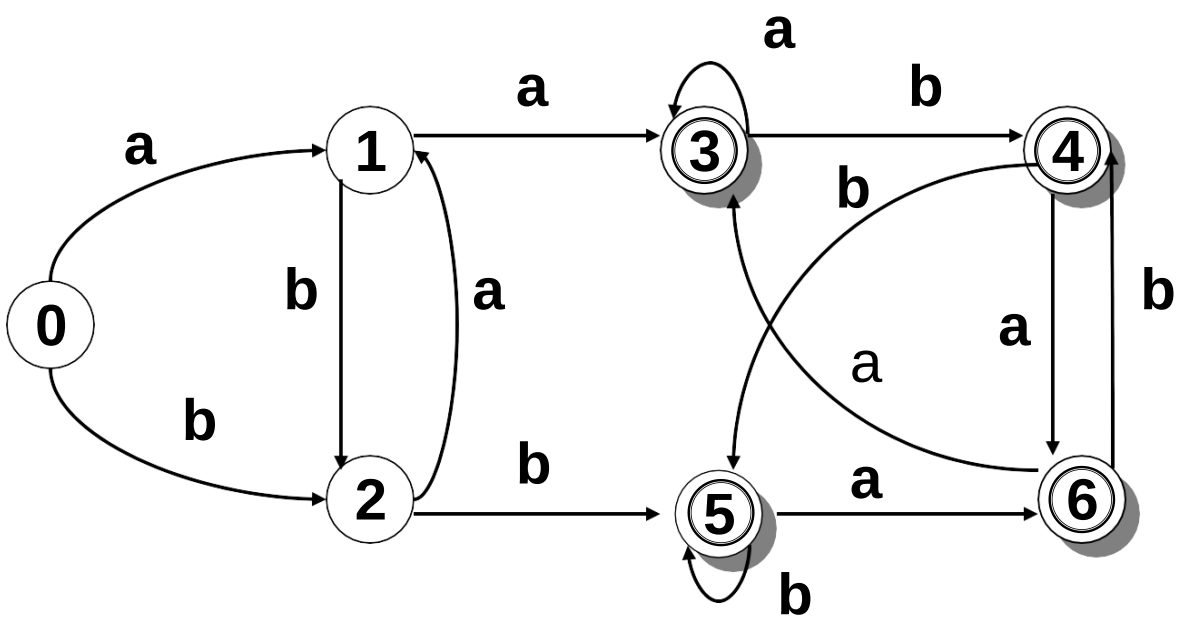
(2) 观察第一个子集，在读入a之后划分为P1=({1,2},{3,4},{5,6,7})

(3) 观察第二个子集，在读入a之后划分为P2=({1,2},{3},{4},{5,6,7})

(4) 观察第四个子集，在读入a之后划分为P3=({1,2},{3},{4},{5},{6,7})

练习

- 简化下面的DFA M



解

1）初始化分P0={{3 , 4, 5, 6},{0, 1, 2}}

2) 观察P0中的第一个子集: {3, 4, 5, 6} $_a$ ={ 3, 6} ⊆ {3, 4, 5, 6} ， {3, 4, 5,6} $_b$ ={4, 5} ⊆ {3, 4, 5, 6}不能再分。{0，1，2} $_a$={1,3} {1,3} 没有完 全包含在 {3,4,5,6}和{ 0,1,2}中，故{0,1,2}一分为二。

1 经a 弧到3，3 ∈{3，4，5，6};

0，2 经a 弧到1，1 ∈{0，1，2}；将{0，1，2}分成{0，2}，{1}

P1={{3 , 4, 5, 6},{0，2}，{1}}

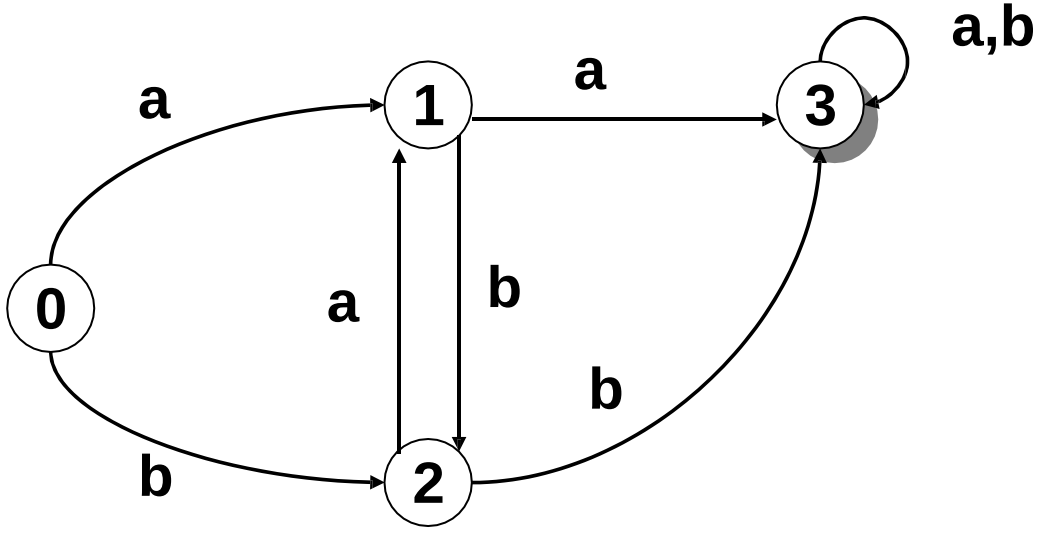
3）检查P1中第二个子集{0，2}：{0，2} $_b$ = {2，5}，{2，5}不完全落在{0，2}，{3，4，5}中，故{0,2}再分成{0}，{2}

得到P2={{0}，{1}，{2}，{3，4，5，6}}

4) 检查P2中的每一个子集，均不可继续划分，故P_{final}=P2={{0}，{1}，{2}，{3，4，5，6}}

解

- 令：3代表{3，4，5，6}，得到：



M = ({0,1,2,3} , {a,b} , f , 0 , {3})

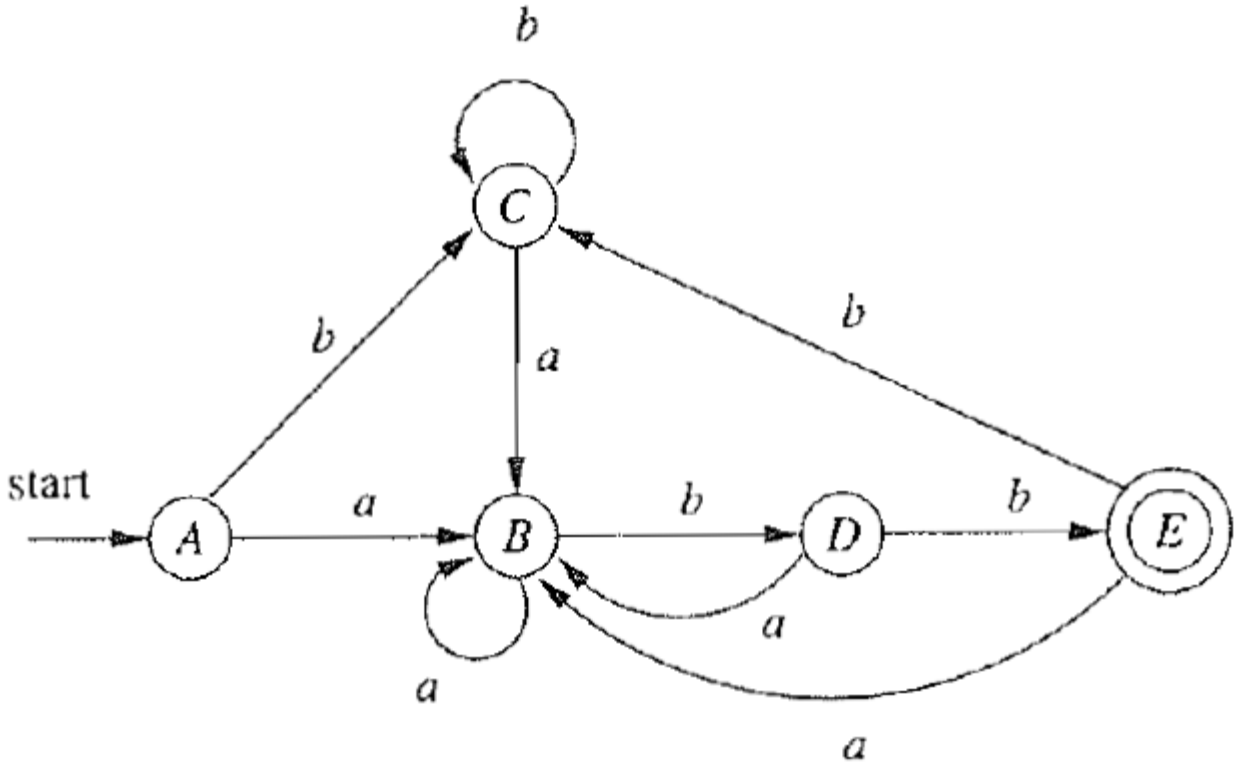
f(0,a) = 1　　f(0,b)=2

f(1,a) = 3　　f(1,b)=2

f(2,a) = 1　　f(2,b)=3

f(3,a) = 3　　f(3,b)=3

练习：最小化下述DFA



总结

- 有限自动机
  - 确定有限自动机 （DFA）
  - 非确定有限自动机 （NFA）
  - NFA确定化
  - DFA最小化

阅读材料：《程序设计语言编译原理（第3版）》，陈火旺等编著，国防工业出版社， 2004年----3.3.1~3.3.3 章节