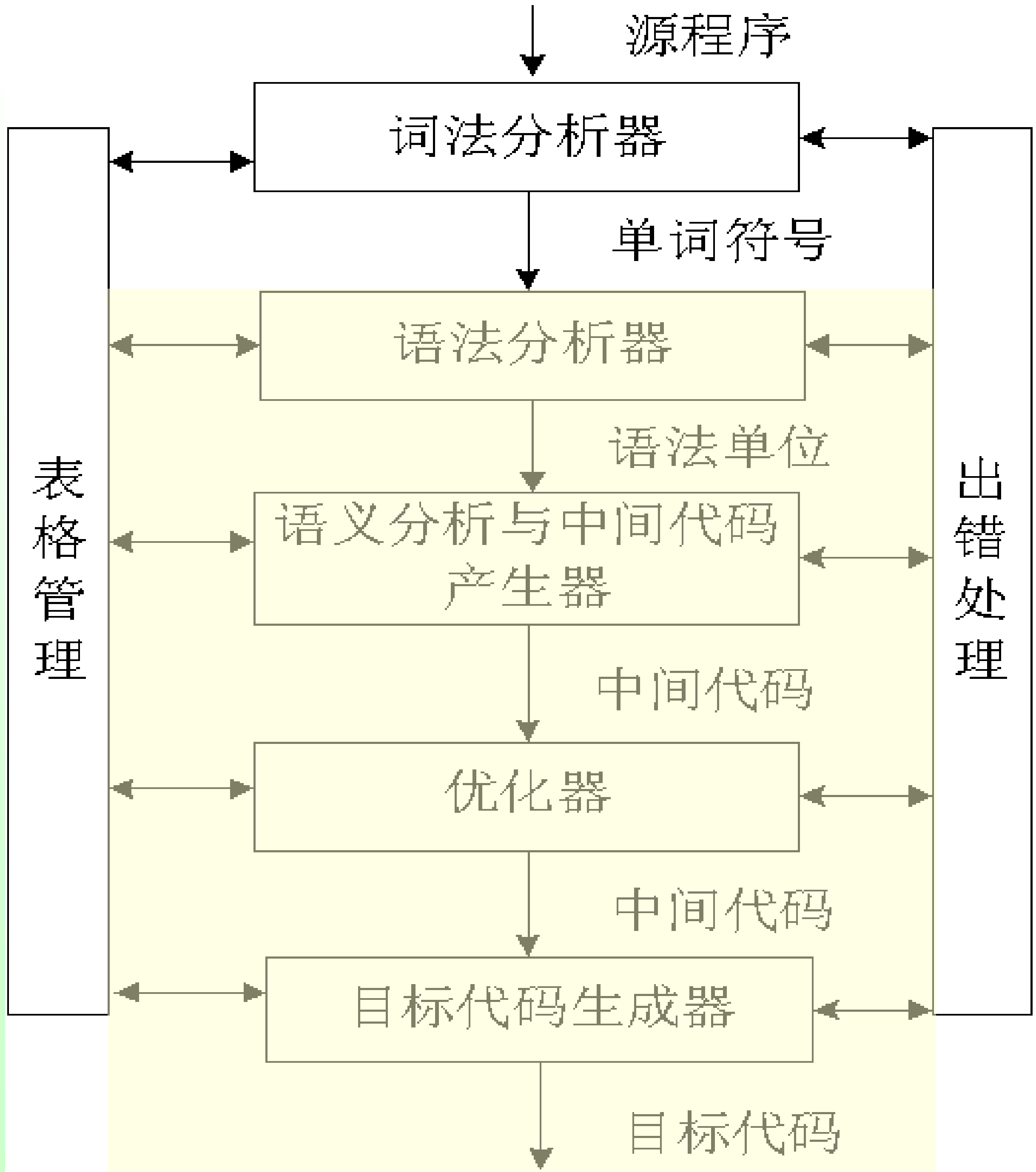
第三章

词法分析

本章要求

- 主要内容：词法分析的任务，手工实现词法分析程序，正规式与有穷自动机，词法分析程序的自动生成
- 重点掌握：词法分析器的功能和接口，用状态转换图设计和实现词法分析程序，正规文法、正规式和有穷状态自动机的概念及相互转换

词法分析程序所处的位置:



3.1 词法分析器的功能

- 功能：

  – 逐个读入源程序字符并按照构词规则切分成一系列单词

- 主要任务：
  – 读入源程序，输出单词符号
- 其他任务：
  – 滤掉空格，跳过注释、换行符
  – 追踪换行标志，指出源程序出错的行列位置
  – 宏展开，……
- 关键：找出单词的分隔符

- **单词：** 是语言中具有独立意义的最小单位，常用单词分类：
  - 保留字：具有固定意义的标识符
  - 运算符
  - 界符
  - 标识符：表示各种名字
  - 常数
- 对于一个程序设计语言，保留字、运算符和界符都是确定的，可以给以固定的编号(种别码)。
- 标识符是根据构词规则定义的，常数是符合定义的各种类型的常数

- 种别码：是对能识别的单词的分类编码

  有多种编码方式：

  - 标识符一般统一为一种：一个编号
  - 常数按类型分别编码：整数、实数、布尔、字符
  - 关键字一般一字一种
  - 运算符一般一符一种
  - 界符一般一符一种

某语言单词的种别码定义举例

单词	种别码	单词	种别码	单词	种别码
and	1	procedure	21	*	41
array	2	program	22	*/	42
begin	3	read	23	+	43
bool	4	real	24	，	44
call	5	repeat	25	—	45
case	6	set	26	、	46
char	7	then	27	. .	47
constant	8	to	28	/	48
do	9	true	29	/*	49
else	10	until	30	：	50
end	11	var	31	:=	51
false	12	while	32	；	52
for	13	write	33	<	53
if	14	标识符	34	<=	54
input	15	整常数	35	<>	55
integer	16	实常数	36	=	56
not	17	字符常数	37	>	57
of	18	‘	38	>=	58
or	19	(	39	[	59
output	20	)	40	]	60

词法分析器的输出

- 1. Token串: 输出源文件中各个有用的单词
  - 格式：(单词的种别码，单词符号的属性值)
  - 单词种别：是对能识别的单词的分类编码(P42)
  - 单词符号的属性值：单词的某种特性或特征
    - **常数的值，标识符的名字等**
    - **保留字、运算符、分界符的属性值可以省略**
  - 文件存放最好有格式，如每个单词占一行方便“语法分析”程序调用
  - P38 例

源程序	token文件

```
{this is a sample program
    writing in simple language}
program example1;
{used for illustrating
    compiling process}
var
    a,b,c:integer;
    x:char;
begin
if (a+c*3 > b) and (b>3) then
    c:=3;
end.
```

举例

```
program
example1
;
var
a
,
b
,
c
:
integer
;
x
:
char
;
begin
if
(
a
+
c
*
3
>
b
)
and
(
b
>
3
)
then
c
:=
3
;
end
.
```

注意token文件的格式

- 2. 符号表
  - 各种常数和标识符一般放在符号表中，在输出的token文件中的单词属性值则存放单词在符号表中的指针
  - 符号表的格式：字符串 if (a>b2) test:=3;

格式1：(数组)

入口	单词名及长度		类型	种属	值	内存地址
1	a	1	整	简单变量	未知	未知
2	b2	2	整	简单变量	未知	未知
3	test	4	实	简单变量	未知	未知
……						

- 格式2：(用指针)

sym_table

入口	单词名字及长度			类型	种属	值	内存地址
1			1	整	简单变量	未知	未知
2			2	整	简单变量	未知	未知
3			4	实	简单变量	未知	未知
……							

a	\0	b	2	\0	t	e	s	t	\0	....

lex

源程序

{this is a sample program writing in simple language}

program example1;

{used for illustrating compiling process}

var

a,b,c:integer;

x:char;

begin

if (a+c*3 > b) and (b>3) then c:=3;

End.

举例

符号表

num	token	name	type	kind	val	addr
(1	24	example1)				
(2	34	a)				
(3	34	b)				
(4	34	c)				
(5	34	x)				
(6	35	3)				

- 3. 其它输出：错误信息和源程序清单
  - 错误信息应该详细，准确，指出出错的具体行、列位置，发生了哪类错误等，方便用户修改

错误处理

- 应尽可能发现更多的错误
- 处理方式
  - 每个程序段单独处理错误
  - 统一处理错误（商用软件系统）
    - 记录式的文件
    - 数据库
- 统计表明，现代软件系统中， 75%的程序代码都是用于处理错误与错误信息
- 商业系统中错误处理的特点是：统一错误编号，编制文档指出错误信息的含义、应对措施、解决方案

词法错误类型

- 非法字符
- 单词拼写错误
- 难以发现下面的错误

  fi (a = x) …

- 在实数是**a.b**格式下，可以发现下面的错误

  123.

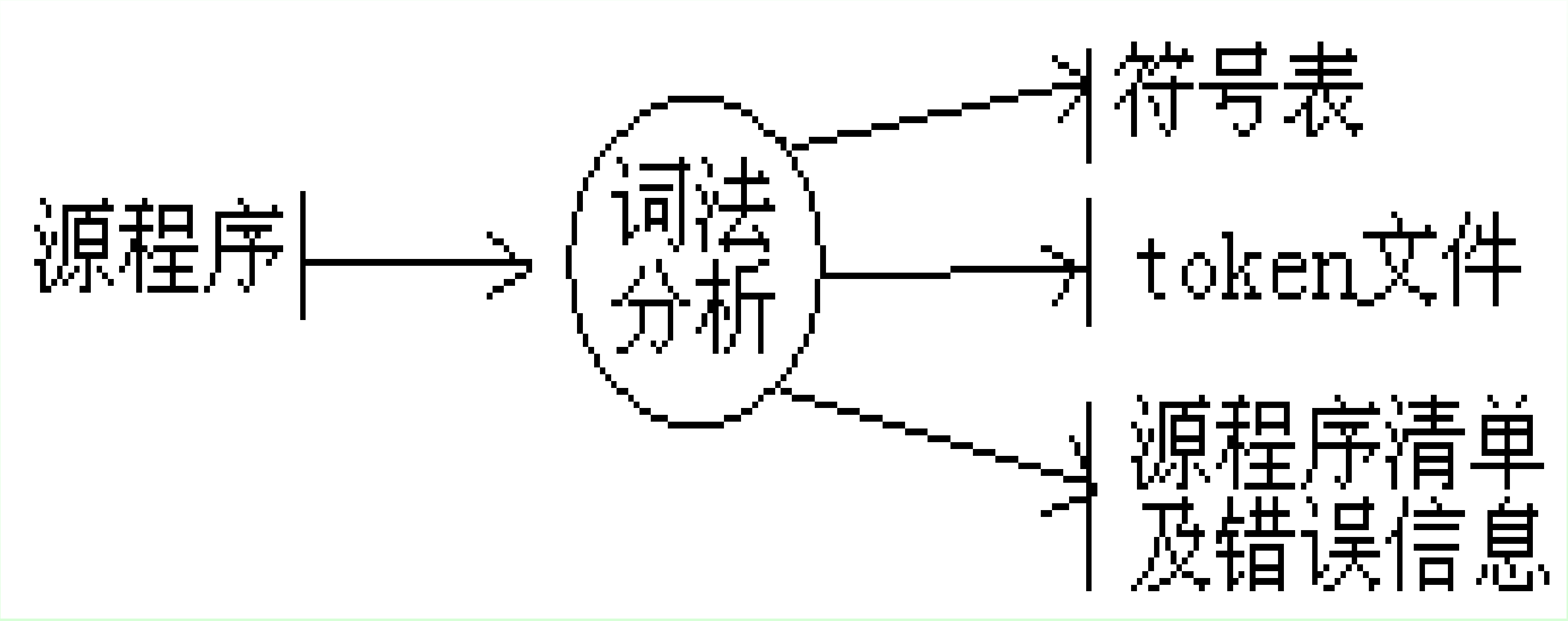
- 词法分析是编译过程中的一个阶段，在语法分析前进行。可以作为一个独立的子程序，独立出来的原因：
  - 简化设计
  - 改进编译效率
  - 增加编译系统的可移植性
- 可以和语法分析结合在一起作为一遍，由语法分析程序调用词法分析程序来获得当前单词供语法分析使用。

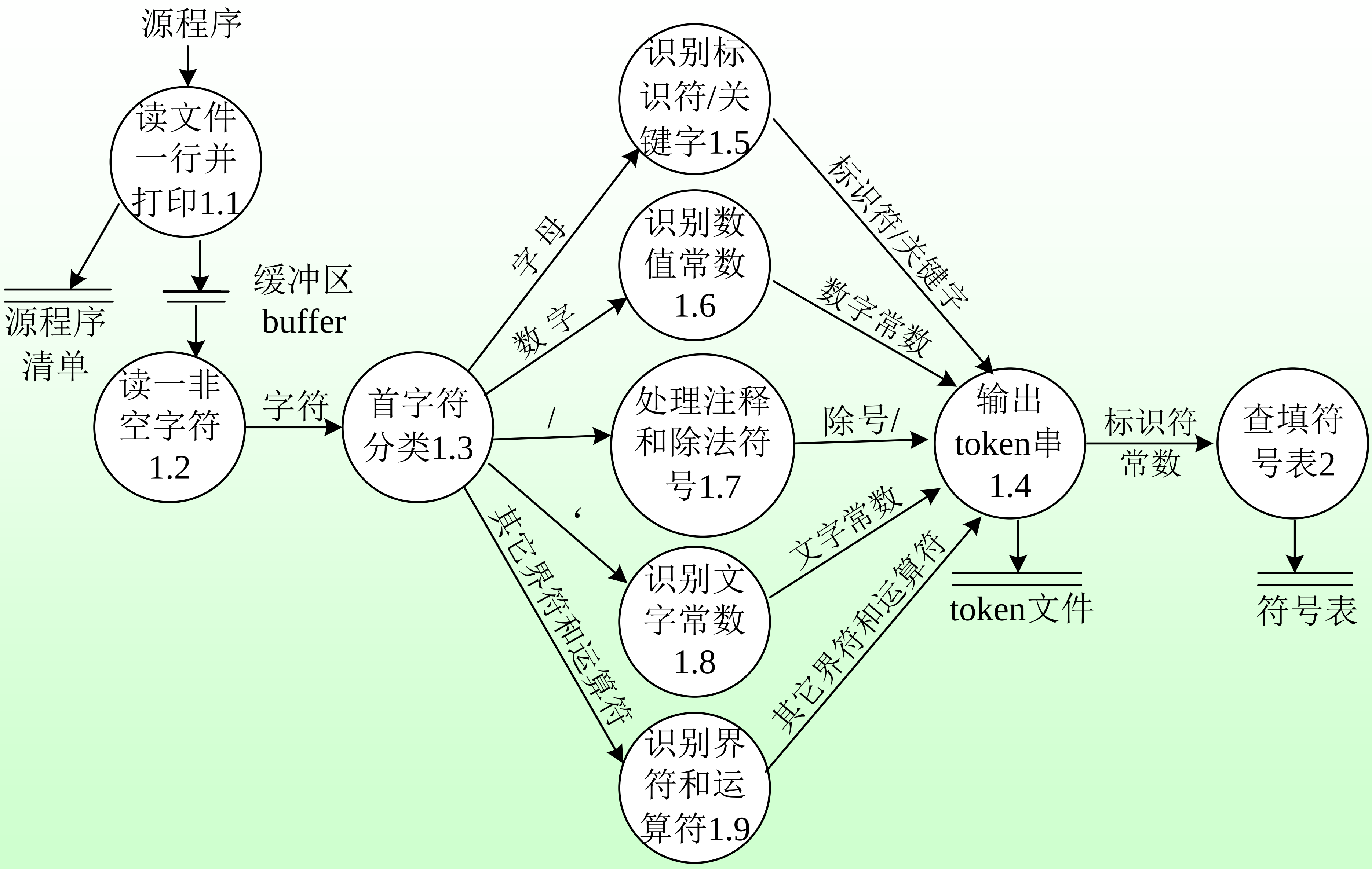
3.2 词法分析程序的设计

扫描器的任务

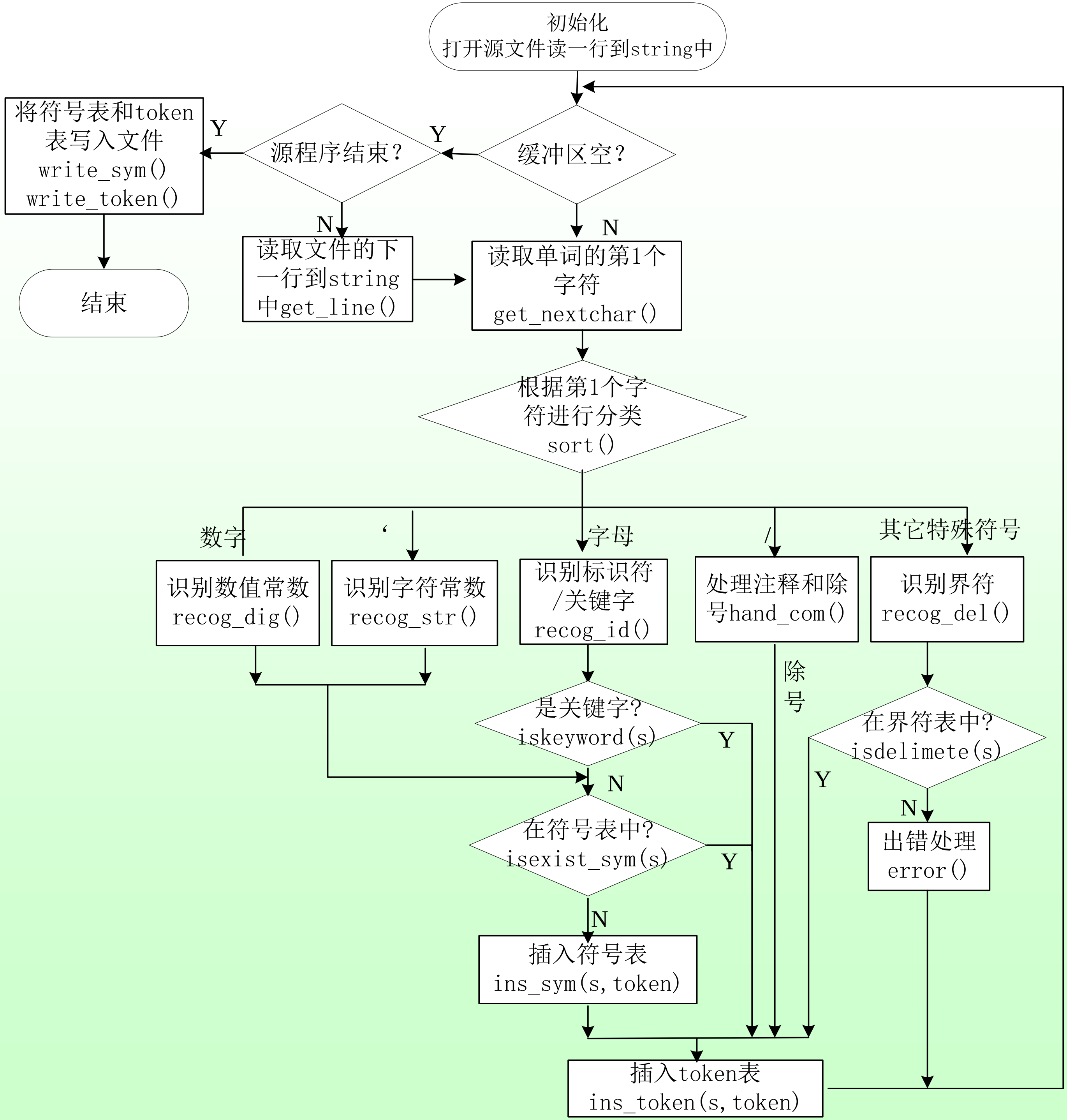
- 组织源程序的输入；
- 按规则拼单词，并转换成二元式形式；
- 删除注解行、空格及无用符号；
- 行计数、列计数；
- 列表打印源程序；
- 发现并定位词法错误；
- 如需要，还要建立关键字表、符号表、常数表等表格。

词法分析程序的接口





源程序
读文件一行并打印1.1
源程序清单
缓冲区buffer
读一非空字符1.2
字符
首字符分类1.3
字母
数字
/
‘
其它界符和运算符
识别标识符/关键字1.5
识别数值常数1.6
处理注释和除法符号1.7
识别文字常数1.8
识别界符和运算符1.9
标识符/关键字
数字常数
除号/
文字常数
其它界符和运算符
输出token串1.4
token文件
标识符常数
查填符号表2
符号表



初始化
打开源文件读一行到string中
缓冲区空？
源程序结束？
将符号表和token
表写入文件
write_sym()
write_token()
结束
读取文件的下
一行到string
中get_line()
读取单词的第1个
字符
get_nextchar()
根据第1个字
符进行分类
sort()
数字
‘
字母
/
其它特殊符号
识别数值常数
recog_dig()
识别字符常数
recog_str()
识别标识符
/关键字
recog_id()
处理注释和除
号hand_com()
识别界符
recog_del()
除
号
是关键字?
iskeyword(s)
在符号表中?
isexist_sym(s)
在界符表中?
isdelimete(s)
出错处理
error()
插入符号表
ins_sym(s,token)
插入token表
ins_token(s,token)
Y
N

- 识别单词前作如下假定：
  - 关键字就是保留字
  - 单词中间不能有分界符(如空格、空白、界符和算符等)
  - 单词中间不能有注释
  - 单词必须在一行内写完，换行后认为是另一个单词
  - 一个单词不能超过规定长度

识别单词

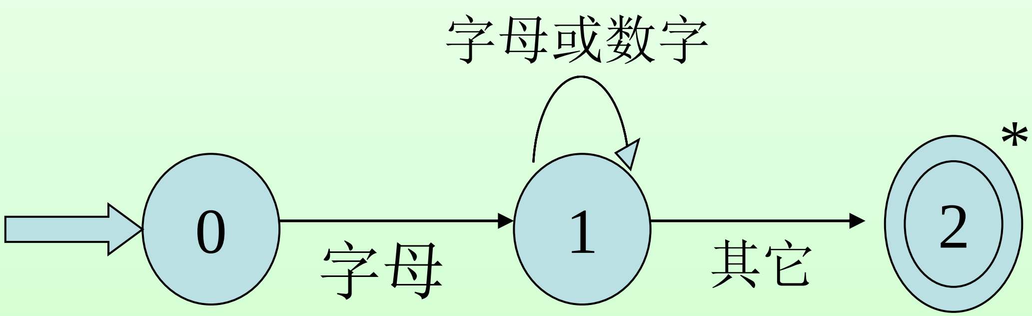
超前搜索技术：如在读取/*　*/时，当读到/时，如何判别是注释还是除法运算？

识别单词：掌握**单词的构成规则**很重要

- 主要包括如下几种单词的识别：
  – 标识符的识别：字母+(字母/数字)
  – 关键字的识别：与标识符相同，最后查表
  – 常数的识别
  – 界符和算符的识别

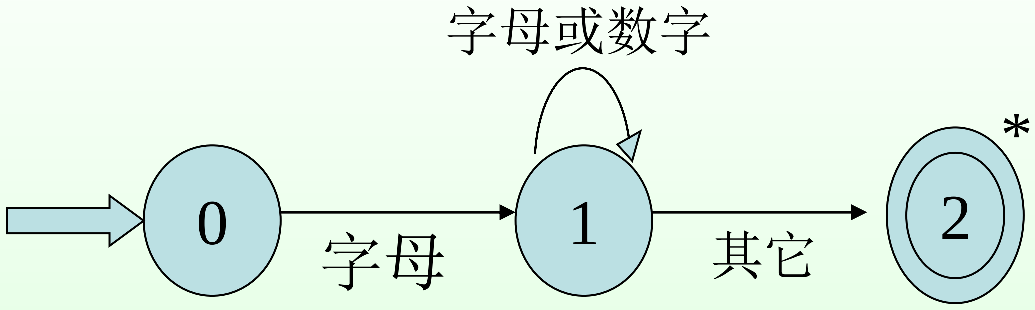
单词的构成规则用状态转换图表示

状态转换图是一张有限方向图。有限个状态，用结点表示状态，其中**有一个初态**(初态用箭头指出)，**至少有一个终态**(终态用双圈表示)。状态之间用带箭头的弧线连结，弧线上标记的字符表示在射出状态下可能出现的输入字符或字符类。



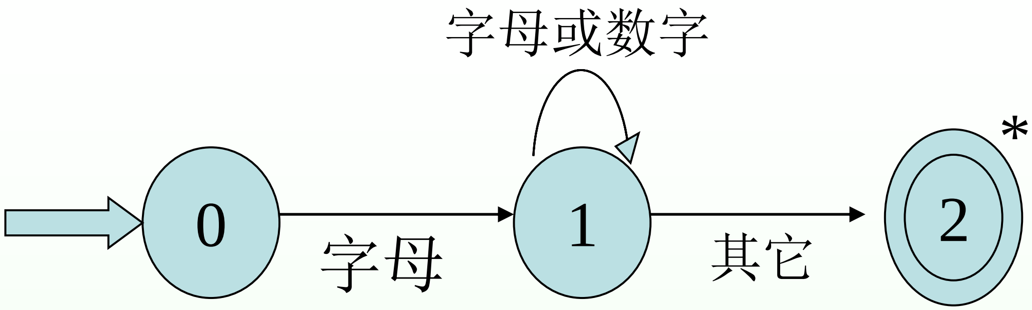
识别标识符的转换图

一个状态图可用于识别一定的字符串，大多数程序设计语言的单词符号都可以用转换图来识别。

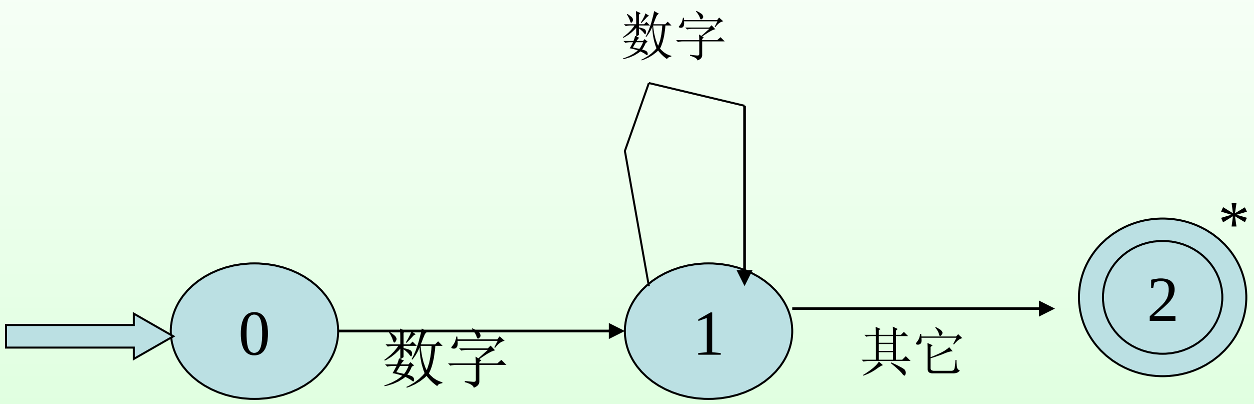


识别过程是：从初始状态0开始，若读入一个字母，转入1状态，若再读入字母或数字，仍处于1状态，否则转向2状态，结束一个标识符的识别过程。状态上的*表示多读入一个符号。

写成C语言的函数形式：




```
recog_id()
{   …
   char state = 0;
   ch = getch();
   do {
       switch(state){
           Case 0: if ch 是字母 state = 1; ch = getch();break;
           Case 1: if ch 是字母或数字 {
                                 state = 1; ch = getch();  }
                         else state = 2;
                         break;
            }
     } while (state != 2);
     回退一个符号。
}
```

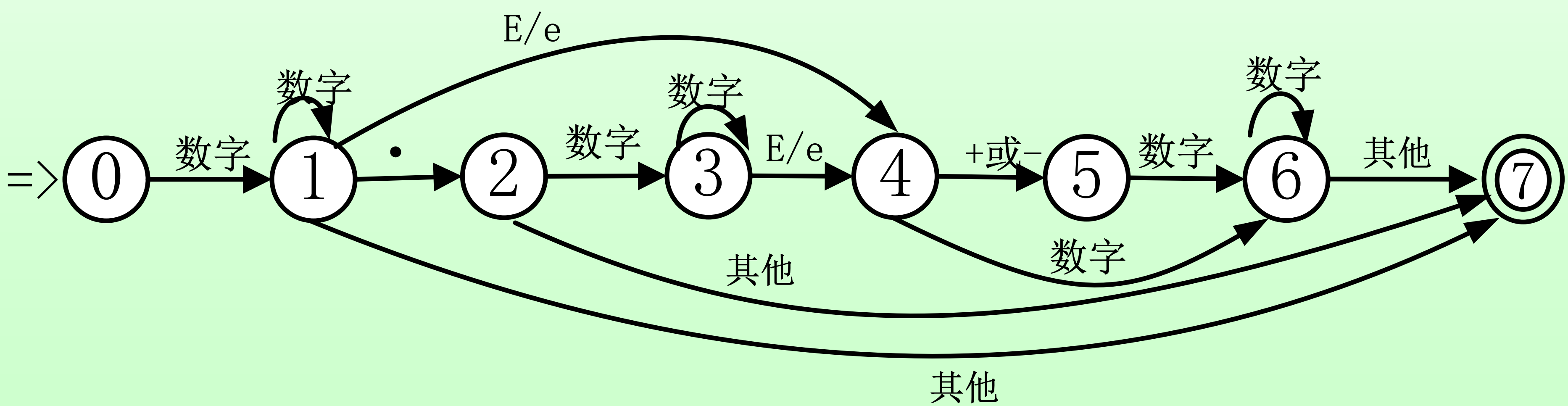


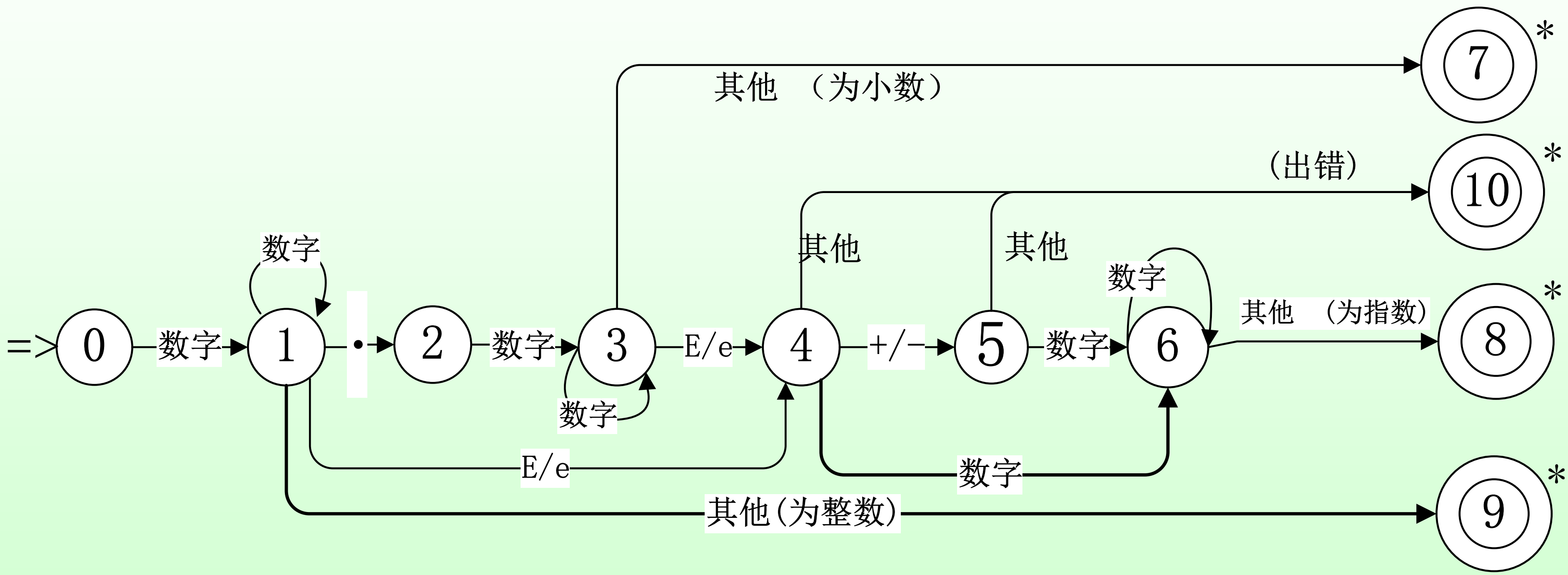
识别整数的转换图

练　习　1

- 画出Pascal中无符号实数的状态转换图 (不带正负号，可表示整数、可表示小数，可带指数部分)
- 如：下面几个数应该是符合规则的数：

  3，3.51，34E3,34.5E2,34.5E+2,34.5E-2

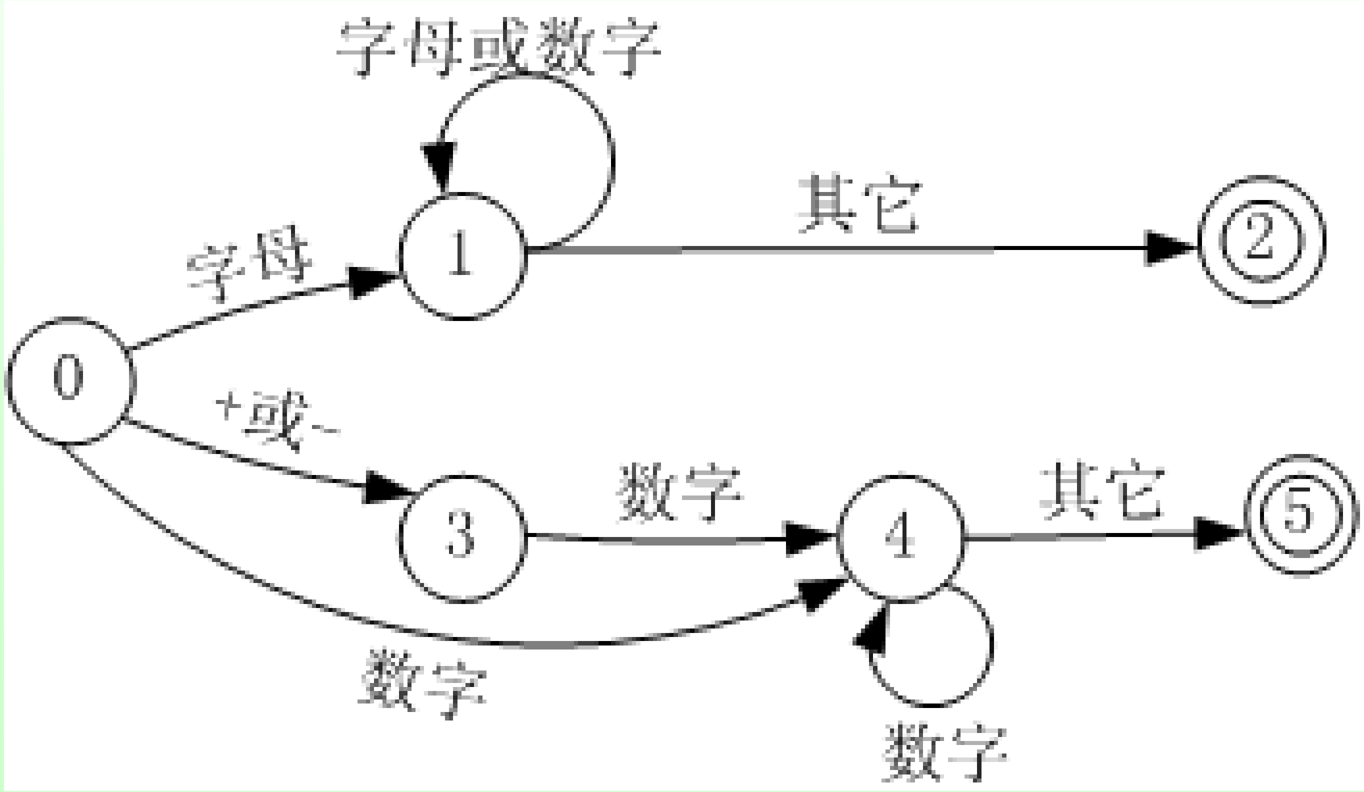




=>
0
数字
1
数字
·
2
数字
3
数字
E/e
4
+/-
5
数字
6
数字
E/e
数字
其他 （为小数）
7
*
（出错）
10
*
其他
其他
其他 （为指数）
8
*
其他(为整数)
9
*

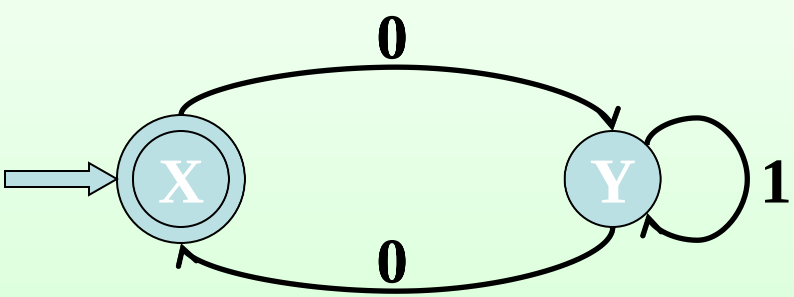
练　习　2

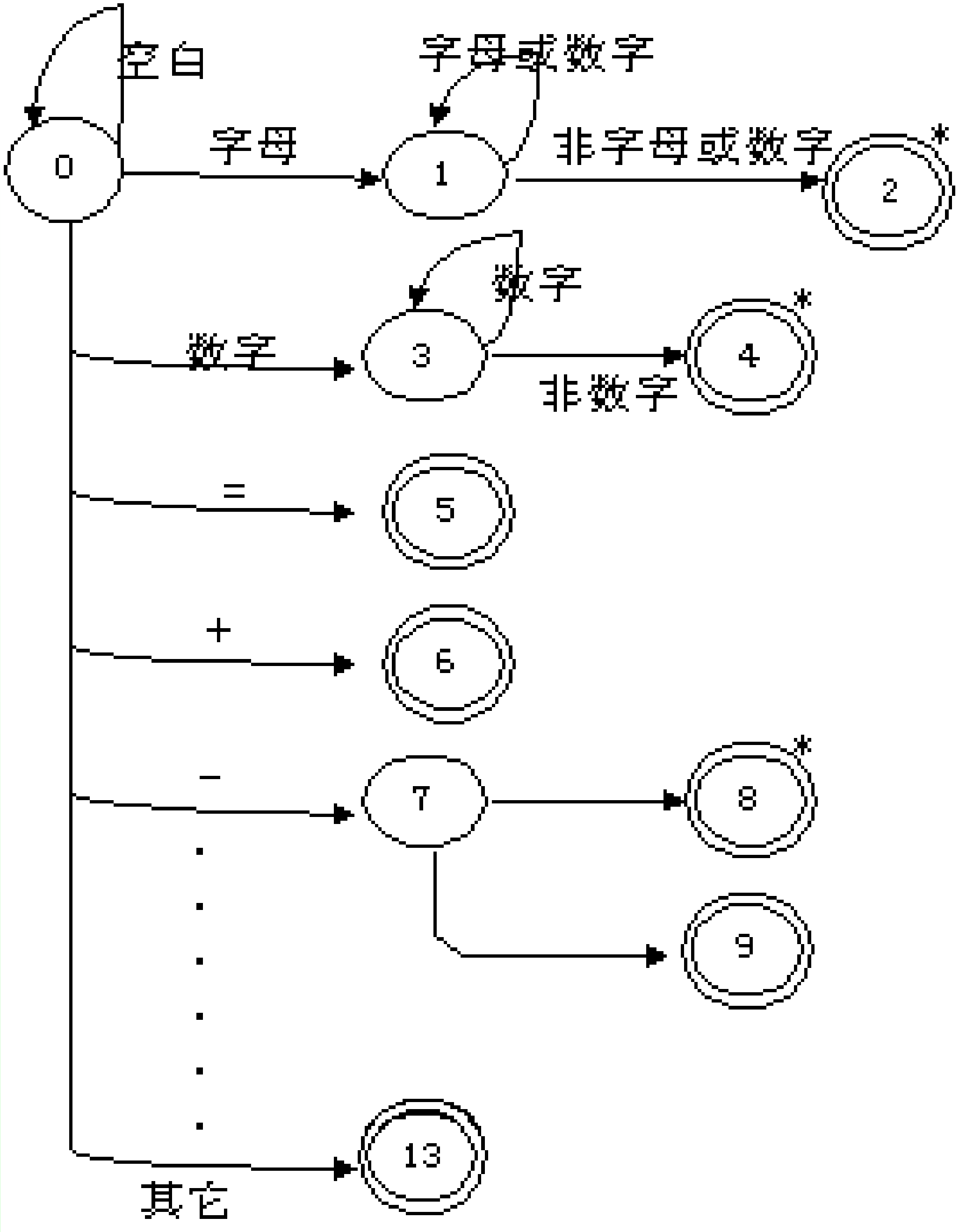
- 画出识别标识符和整常数(可带正负号)的状态转换图



练　习　3

- 以下状态转换图接受的字符集合是什么？





对简单语言进行词法分析的状态转换图

某简单语言的词法分析程序的实现

状态转换图的实现：使用一个switch case 语句：每条分支对应一个case语句段

见书P45 例

词法分析器的自动生成

- 正规式
- 正规文法
- 有穷自动机

3.3 正规文法、正规式与有限自动机

- 本节要求

1　能根据自然语言描述构造NFA
2　掌握NFA转换为DFA，DFA的化简
3　掌握正规文法、正规式和有穷自动机间的转换

- 为了讨论词法分析程序的自动生成问题，将状态转换图加以形式化。

一、正规文法

- **正规文法**：文法G=(V_N，V_T，P，S)中的每个产生式的形式都是A→aB或A→a，其中A和B都是非终结符，a是终结符串。

下面定义的标识符和无符号整数都是正规文法：

<标识符>→letter | letter〈字母数字〉

<字母数字>→letter | digit | letter〈字母数字〉
| digit〈字母数字〉

<无符号整数>→digit | digit〈无符号整数〉

- **结论：** 每一种程序设计语言，都有它自己的字符集V，语言中的每一个单词或者是V上的单个字符，或者是V上的字符按一定方式组成的字符串。组成方式就是对字符或字符串进行（连接）“．”、或“｜”(并)、或“＊／＋”闭包运算。

二、正规式

- 正规式也称为正则表达式，是表示正规集的工具。
- 正规式（regular expression）是说明单词的pattern的一种重要的表示法，是单词的描述工具。
- 下面是正规式和它所表示的正规集的递归定义

- 正规式和正规集的递归定义：（设字母表为Σ）

1、 ε和Φ都是Σ上的正规式，表示{ε}和{ }；

2 、任何a∈ Σ，则a是正规式，表示{a}；

3 、假定r和s都是Σ上的正规式，分别表示语言

L(r)和L(s)：

a） (r) | (s)是正规式，表示L (r) U L (s) ；

b） (r)(s)是正规式，表示L (r)L (s)；

c） (r)*是正规式，表示(L (r))*；

d） (r)是正规式，表示L (r)；

4、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。

|——或；• ——连接；* ——闭包

规定优先顺序为“*”、“• ”、“|”

(a)|((b)*(c)) ⟶ a|b*c

例1：令Σ={a，b}，Σ上的正规式和相应的正规集有：

正规式	正规集
a	{a}
ba*	所有以b开头后跟任意多个a的串
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 或b组成的串
(a|b)*(aa|bb)(a|b)*	所有含有两个相继的a或两个相继的b的串

程序设计语言的单词都能用正规式来定义.

例2：令Σ={l，d}，l 代表字母,d 代表数字,则Σ上的正规式: $r = l(l \mid d)^*$ 定义的正规集为: {l,ll,ld,lll,ldd,……},就是Pascal和 多数程序设计语言允许的的标识符的词法规则。

例3：令Σ＝{d，.，e，＋，－}，其中d为0～9中的数字。

则Σ上的正规式: $d^*(.dd^* \mid \varepsilon)(e(+ \mid - \mid \varepsilon)dd^* \mid \varepsilon)$

表示PASCAL语言中的无符号实数。

比如：2, 12.59, 3.6e2, 471.88e-1等都是正规式表示集合中的元素。

练　习

1、Σ={a,b}，则Σ上所有以b开头，后跟若干个ab的字的全体所对应的正规式。

$$b(ab)^*$$

2、 Σ={a,b}，写出不以a开头，但以aa结尾的字符串集合的正规式。

$$b(a|b)^*aa$$

- 思考题：

  Σ={d，.}，则Σ上表示无符号数的正规式是什么？（不考虑含e的科学计数法，其中d为0~9的数字）

  如：2，12.59，471.88等都是该集合中的元素。

$$dd^*(.dd^* \mid \varepsilon)$$

正规式的等价

- 若两个正规式e_1和e_2所表示的正规集相同,则e_1和e_2**等价**,写作$e_1=e_2$。
- 设r,s,t为正规式，正规式服从的代数规律有：
  - 1. $r|s=s|r$ “或”服从交换律
  - 2. $r|(s|t)=(r|s)|t$ “或”的可结合律
  - 3. $(rs)t=r(st)$ “连接”的可结合律
  - 4. $r(s|t)=rs|rt$
    $(s|t)r=sr|tr$ 分配律
  - 5. $\varepsilon r=r\varepsilon=r$ ε是“连接”的恒等元素 零一律
  - 6. $e^*=e^+|\varepsilon$
  - 7. $e^+=e^*e=ee^*$
  - 8. $(e^*)^*=e^*$

三、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，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DFA

一个确定的有穷自动机(DFA) M是一个五元组：
$M=(S, \Sigma, \delta, s_0, F)$，其中：

1. S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；
3. δ是转换函数，是在$S\times\Sigma\rightarrow S$上的单值映射，$\delta(s, a)=s'$ $(s\in S, s'\in S)$，就意味着，当前状态为s，输入符为a时，将转换为下一个状态s'，我们把s'称作s的一个后继状态；
4. $s_0\in S$是唯一的一个初态；
5. $F\subseteq S$是一个终态集(可空)，也称可接受状态或结束状态。

DFA的矩阵表示

例3：有DFA M =({0,1,2,3},{a,b}, δ,0,{3})

δ为：

δ(0,a) = 1　δ(0,b) = 2

δ(1,a) = 3　δ(1,b) = 2

δ(2,a) = 1　δ(2,b) = 3

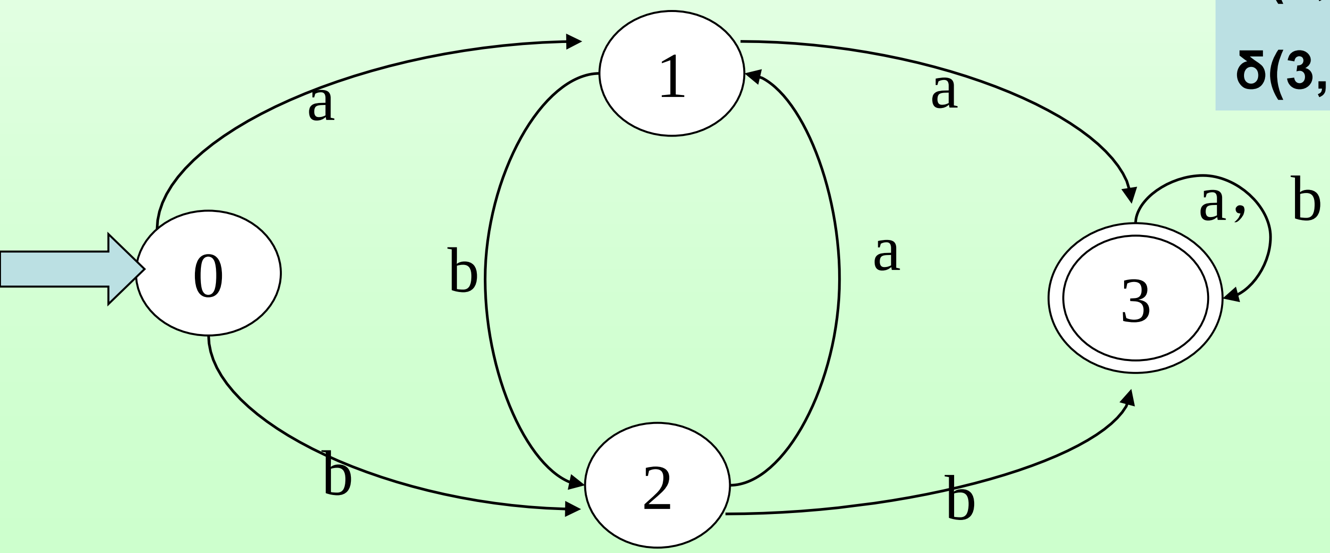
δ(3,a) = 3　δ(3,b) = 3

行表示状态，列表示输入字符，矩阵元素表示δ (s，a)的值，称为**状态转换矩阵**。

状态 \ 输入	a	b
0	1	2
1	3	2
2	1	3
3	3	3

- 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(可以是0个)终态结点，初态结点冠以双箭头“=>”，终态结点用双圈表示，若 $\delta(k_i,a)=k_j$，则从状态结点k_i到状态结点k_j画标记为a的弧。

δ(0,a) = 1	**δ(0,b) = 2**
δ(1,a) = 3	**δ(1,b) = 2**
δ(2,a) = 1	**δ(2,b) = 3**
δ(3,a) = 3	**δ(3,b) = 3**



- DFA的确定性表现在：
  - 对任何状态s ∈S，在读入了输入符号a ∈ Σ 之后，能够唯一地确定下一个状态
  - 映射函数δ：S×Σ→S是一个单值函数
  - 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。

- 字α可为DFA M所接受（识别）：

  对于Σ* 中的任何字α，若存在一条从初态结点到某个终态结点的通路，且这条通路上所有弧的标记符号连接成的字等于α。

- 若M的初态结点又是终态结点，则空字ε可为M所识别。

- DFA M所能识别的符号串的全体记为L(M).

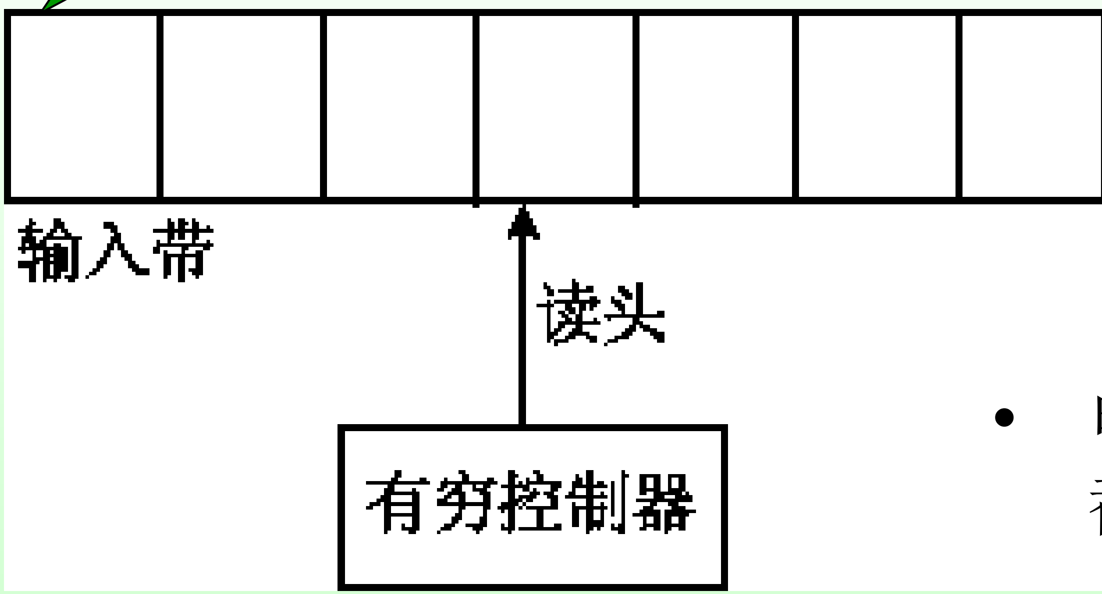
- 对于任何两个有穷自动机M和M′，如果L(M)=L(M′)，则称M与M′是**等价**的.

在
入
准
如

整个模型由
- 输入带：
- 读头： 可
- 有穷控制

如果读头移动到最后一个符号后面，状态正好是终结状态，则称输入带上的符号组成的字能被该有穷自动机所识别

，读
位置，
状态
状态



有穷自动机模型

- 电梯控制系统，人脑都是有穷自动机
- 文本编辑程序有穷状态系统

- 结论：

  $\sum$上一个符号串集V $\subseteq \sum^*$是正规的，当且仅当存在一个$\sum$上的确定有穷自动机M，使得V=L(M)。

文法和自动机的对比

- 文法是语言的生成系统，是从产生的观点来描述语言的。
- 自动机是语言的识别系统，是从识别的观点来描述语言的

不确定的有穷自动机NFA

- 定义：不确定的有穷自动机NFA也是一个五元组，M={S，∑，δ，s_0， F }，其中：

  S为状态的有穷状态集，

  ∑ 为有穷输入字母表，

  δ为S × ∑* 到S的幂集(2^S)的一种映射:

  $$S \times \Sigma^* \rightarrow 2^S$$

  s_0 ⊆S是初始状态集，

  F ⊆S为终止状态集(可空).

NFA的矩阵表示

- 例4：NFA M=({S，P，Z}，{0，1}，δ，{S，P}，{Z})

  其中：

  δ(S，0)={P}

  δ(S，1)={S，Z}

  δ(Z，0)={P}

  δ(Z，1)={P}

  δ(P，1)={Z}

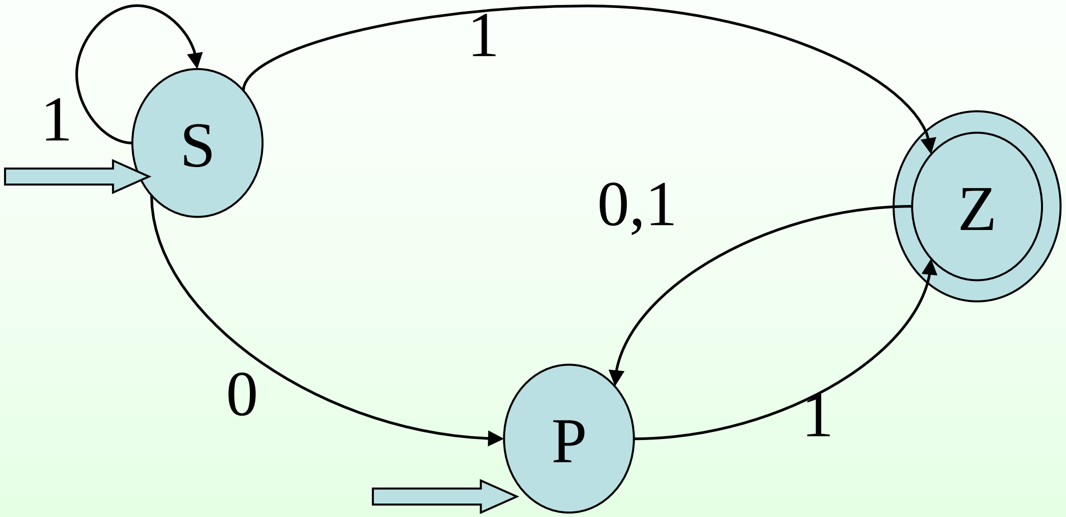
- 矩阵表示

状态 \ 输入	0	1
S	{P}	{S,Z}
P	{}	{Z}
Z	{P}	{P}

从NFA的矩阵表示中可以看出，表项是状态的集合，而在DFA的矩阵表示中，表项是一个状态

状态图表示

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



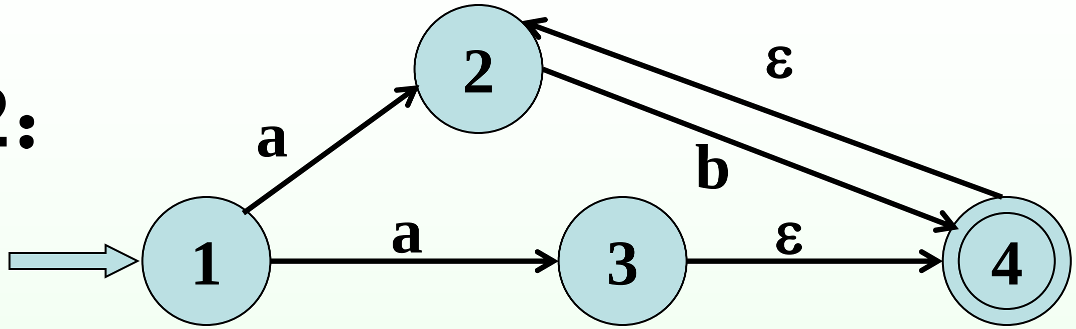
一个含有m个状态，n个输入字符的NFA的状态转换图：有m个结点，每个结点可射出**若干条**弧与别的结点相连接，每条弧用∑* 上的一个字来表示(这些字可以相同，也可以是ε)。

整个图**至少有一个初始结点**以及**若干个(可以是0个)终态结点**，某些结点既可以是初态结点，又可以是终态结点。

Σ^*上的符号串t被NFA M接受（识别）：

- 对于Σ*中的任何一个串t，若存在一条从某一初态结点到某一终态结点的通路，且这条通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的某些结点既是初态结点又是终态结点；或者存在一条从某个初态结点到某个终态结点的道路,其上所有弧的标记均为ε，那么空字ε可为M所接受。

- **例2:**



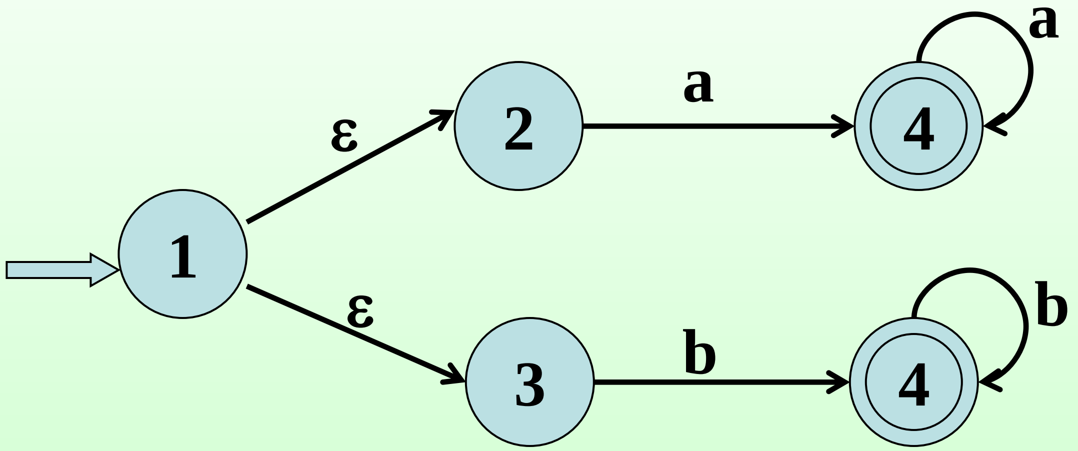
接受串abb的移动序列:

-->$1\xrightarrow{a}2\xrightarrow{b}4\xrightarrow{\varepsilon}2\xrightarrow{b}4$

-->$1\xrightarrow{a}3\xrightarrow{\varepsilon}4\xrightarrow{\varepsilon}2\xrightarrow{b}4\xrightarrow{\varepsilon}2\overset{b}{\text{——}}4$

ε-转换（ ε -transition）:是无需考虑输入串就有可能发生的转换。

例3:下列NFA定义的语言是什么？



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 $\subseteq \Sigma^*$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)。

DFA与NFA的主要区别

- （1）DFA任何状态都没有ε转换，即没有任何状态可以不进行输入符号的匹配就直接进入下一个状态；
- （2）DFA对任何状态*s*和任何输入符号*a*，最多只有一条标记为*a*的边离开*s*，即转换函数δ：$S \times \Sigma \rightarrow S$是一个单值部分函数。
- （3）DFA的初态唯一，NFA的初态为一集合。

NFA的确定化

- 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)。也就是说：对每个NFA N存在着与之等价的DFA M。

- 方法：（子集法）将NFA转换成接受同样语言的DFA。
- NFA确定化的基本思路是： DFA的每一个状态对应NFA的一组状态.

- NFA确定化的基本步骤是：
- **第一步：对NFA的状态图进行改造。**

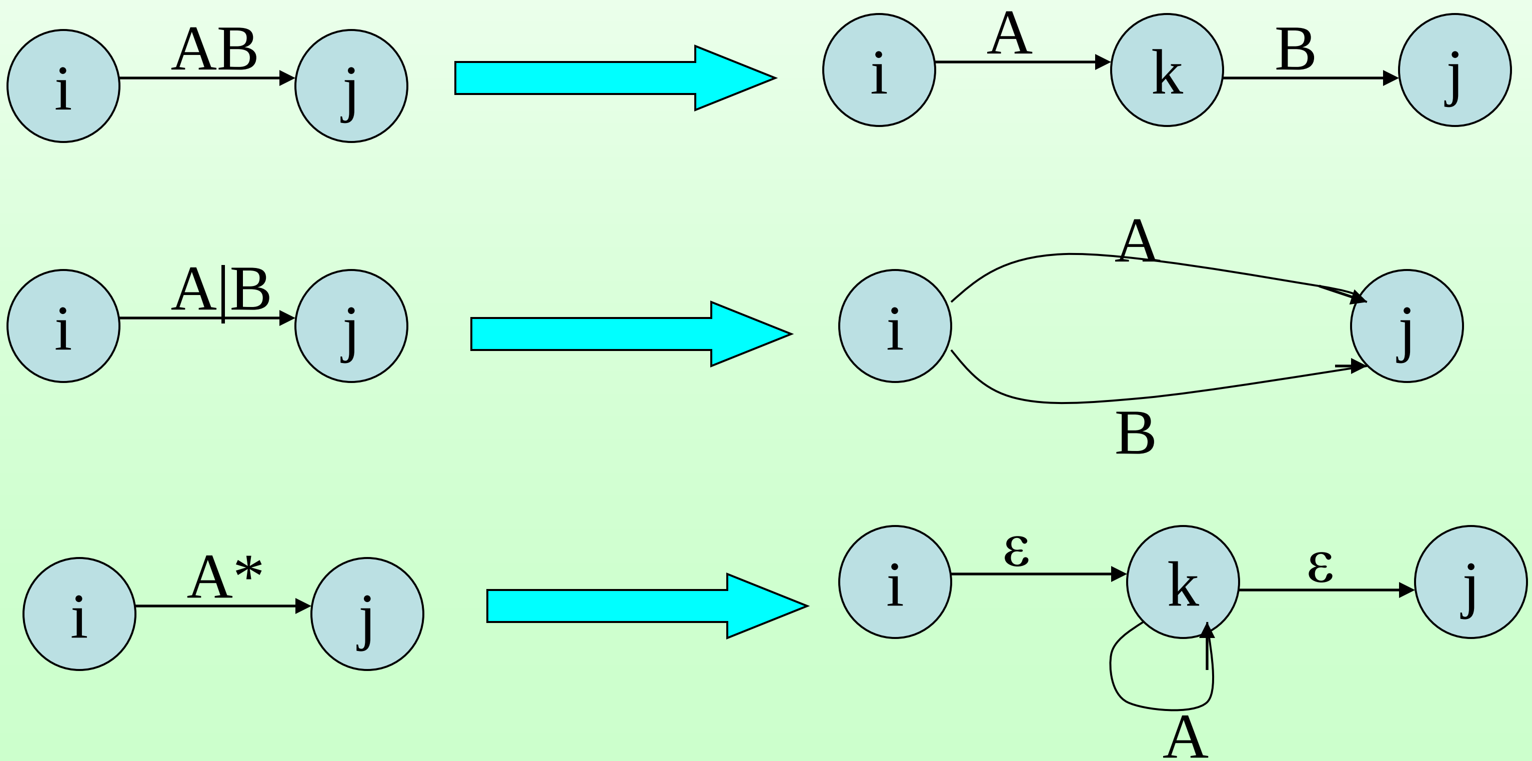
由于NFA可能有多个初态结点、多个终态结点、每条弧上的标记可能是Σ^*上的一个字，因此首先将其改造，使之成为只有一个初态结点、一个终态结点、每条弧上的标记只能是单个输入符号或者ε。

- **第二步：对上述改造后的NFA进行确定化。**

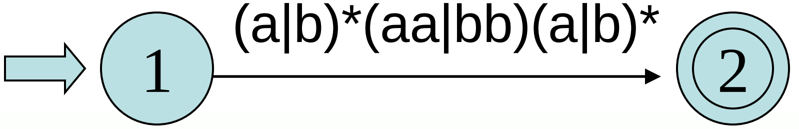
去掉ε。

NFA的确定化

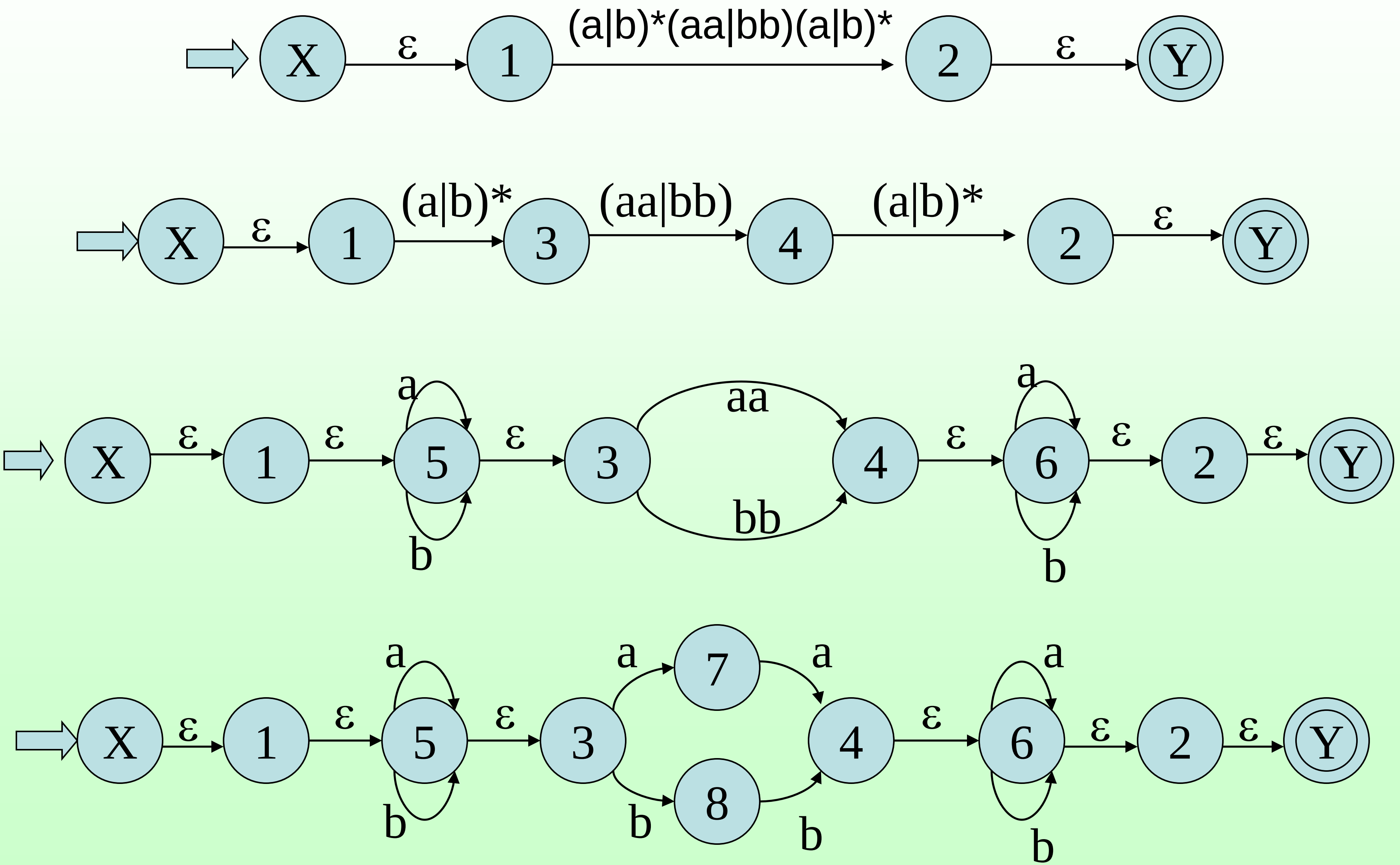
- **第一步：对NFA的状态图进行改造**
  - (1)增加状态X,Y,使之成为新的唯一的初态和终态。从X引 **ε** 弧到原初态结点, 从原终态结点引 **ε** 弧到Y结点。
  - (2) 对状态图进一步进行如下形式的改变



- 例5：有NFA如下：



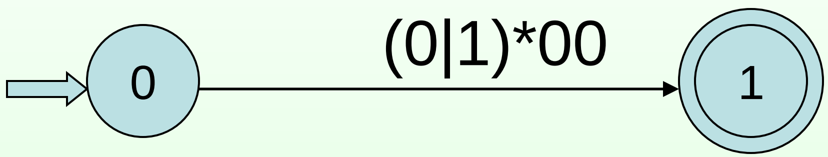
1
(a|b)*(aa|bb)(a|b)*
2

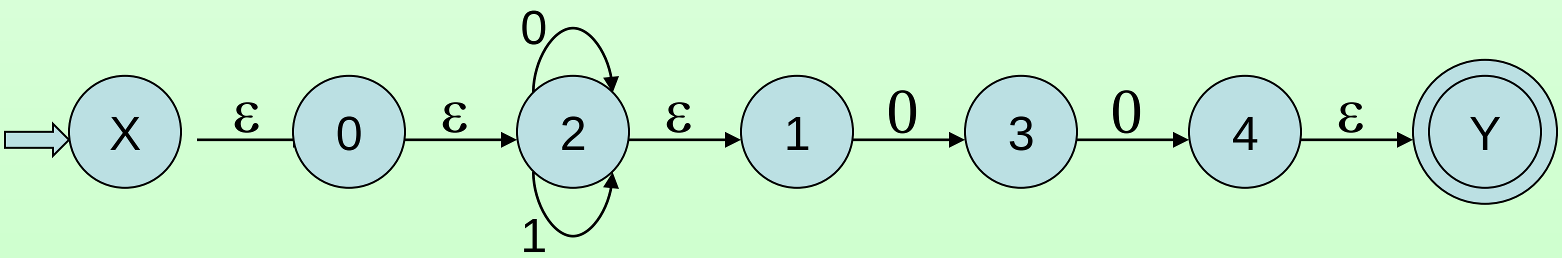


X
ε
1
(a|b)*(aa|bb)(a|b)*
2
ε
Y
X
ε
1
(a|b)*
3
(aa|bb)
4
(a|b)*
2
ε
Y
X
ε
1
ε
5
a
b
ε
3
aa
bb
4
ε
6
a
b
ε
2
ε
Y
X
ε
1
ε
5
a
b
ε
3
a
7
a
b
8
b
4
ε
6
a
b
ε
2
ε
Y

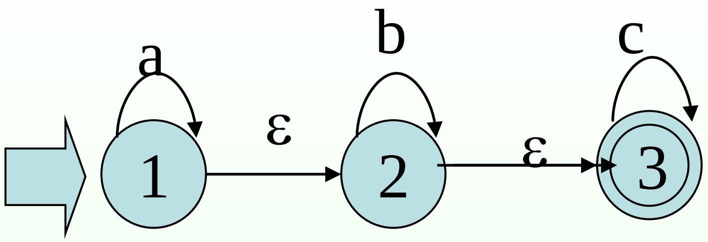
练 习

- 求下述NFA对应的DFA（完成第一步）

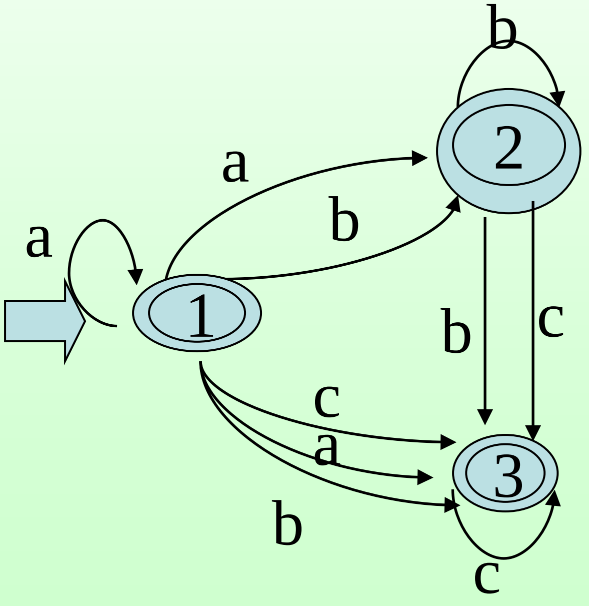




- 上述NFA带有ε弧，称为具有ε转移的不确定的有穷自动机



- 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)。



- 第二步：对上述改造后的NFA进行确定化

❖ ***状态集合I的ε－闭包***ε-closure(I)，是一状态集

  ❖ 任何状态q ∈ I，则q ∈ ε-closure(I)；

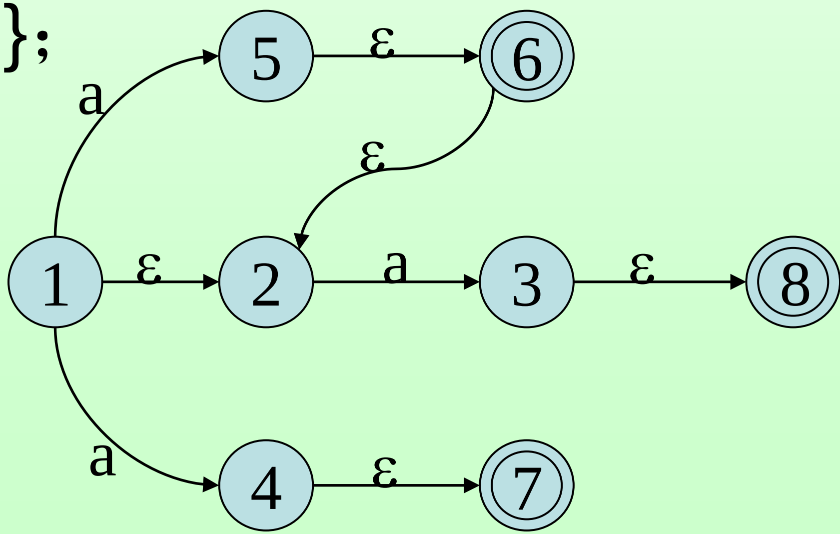
  ❖ 任何状态q ∈ I，则q经任意条ε弧而能到达的状态q′ ∈ ε-closure(I) 。

例： I={1}, ε-closure(I)={1,2}；

I={5}, ε-closure(I)={5,6,2}；

I={5,4}, ε-closure(I)=？

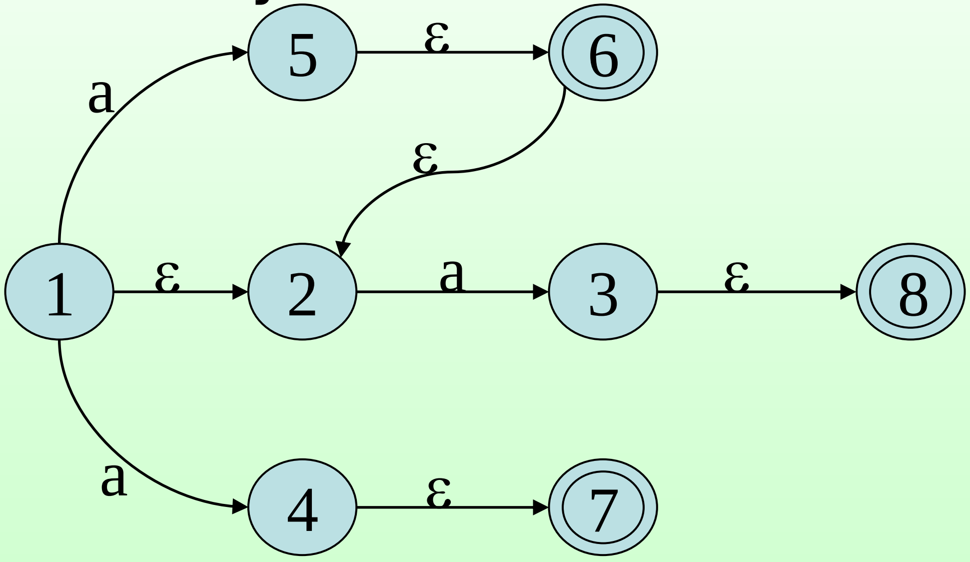
ε-closure(I)={5,4,6,2,7}



❖ 状态集合I的a弧转换，Ia = ε-closure(J) ,其中 **J=move(I,a)**，即所有可从I中的某一状态经过一条a弧而到达的状态的全体。

I={1}, J=move(I,a) = {5, 4} ;

Ia= ε-closure({5, 4})={2,4,5,6,7}



I={1,2}, Ia=?

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

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

❖ 对NFA进行确定化，构造状态转换表：

1. 对$\Sigma = \{a_1 \ldots a_k\}$, 构造一个k+1列的状态转换表，行为状态，列为输入字符，置该表的首行首列为 ε-closure(X)，(X为第一步完成后的唯一的开始状态)。

列1	列2(a_1)	列3(a_2)	…….	列K+1(a_K)
ε-closure(X)			…….	

2. 若某行的第一列的状态已确定为I，则计算第i+1（i=1,2,…,k）列的值为Ia_i。

列1	列2(a_1)	列3(a_2)	…….	列K+1(a_K)
ε-closure(X)	Ia_1	Ia_2	……	Ia_k

3. 检查第2步所创建的该行上的所有状态子集，看它是否已在第一列出现，若未出现，将其添加到后面的空行上作为新的一行。

I	I_{a_1}	I_{a_2}	……	I_{a_k}
ε-closure(X)			……	
I_{a_1} ?				
I_{a_2} ?				
……?				
I_{a_k} ?				

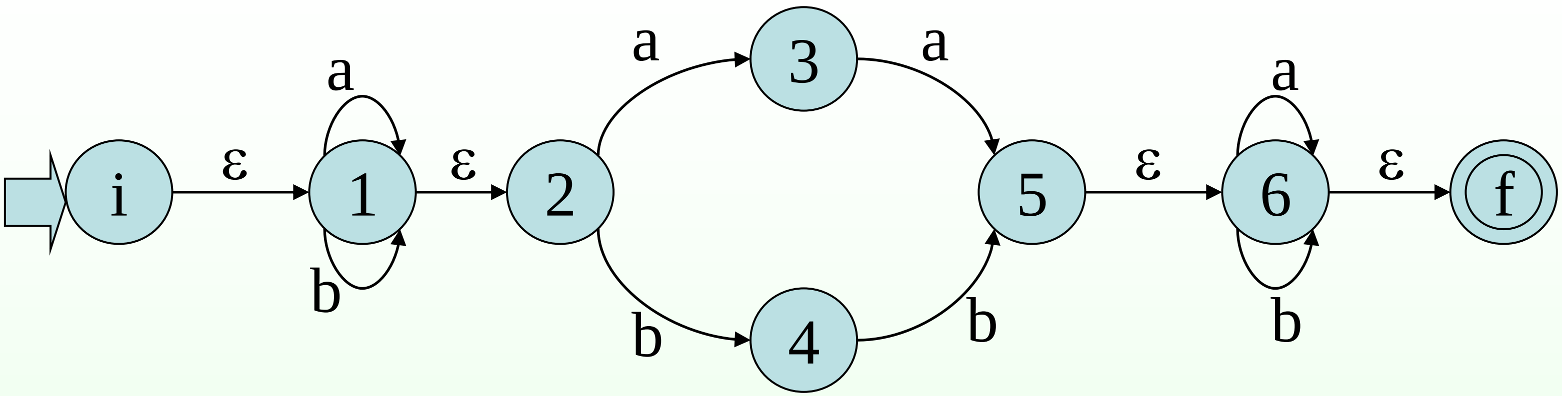
4. 重复步骤2，3，直到状态不再增加，即所有状态子集均在第一列中出现。

I	I_{a_1}	I_{a_2}	……	I_{a_k}
ε-closure(X)			……	
?			……	
?			……	
……?				

5. 将每个状态子集视为一个新的状态，就得到一个确定的有穷自动机，初态就是首行首列的状态，终态是含有原有终态的所有状态。

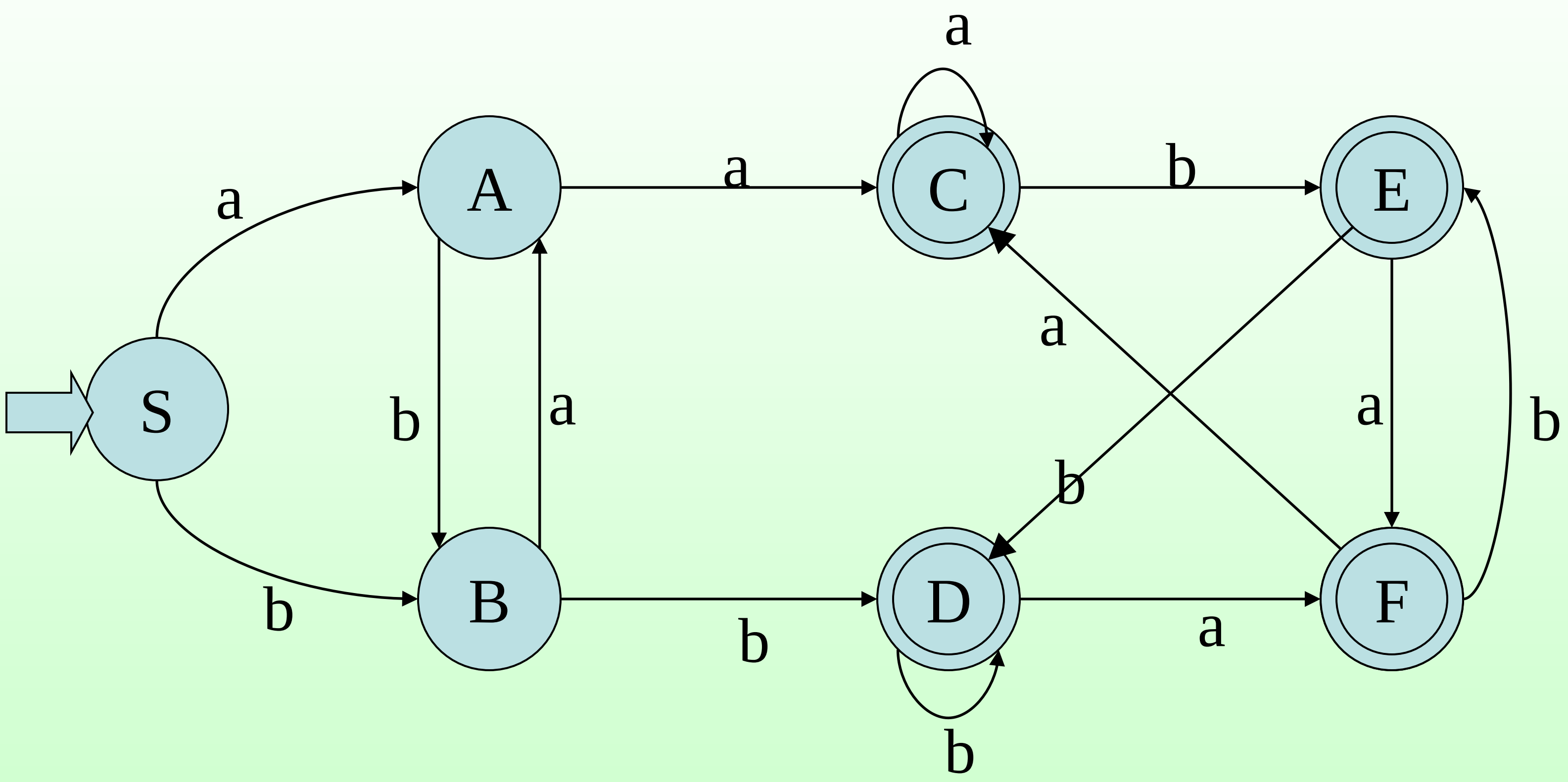
状态	a_1	a_2	....	a_k
S	**A**	**B**	......	**A**
A	B	C	......	D
B	C	A	......	B
C	C	A	......	C
D	E	B	......	B
E	D	A	......	F
F	A	B	......	C

例6：将下述NFA确定化



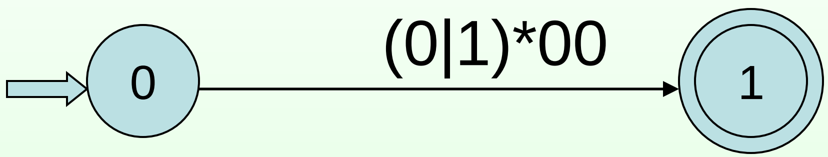
I		I_a		I_b	
{i,1,2}	S	{1,2,3}	A	{1,2,4}	B
{1,2,3}	A	{1,2,3,5,6,f}	C	{1,2,4}	B
{1,2,4}	B	{1,2,3}	A	{1,2,4,5,6,f}	D
{1,2,3,5,6,f}	C	{1,2,3,5,6,f}	C	{1,2,4,6,f}	E
{1,2,4,5,6,f}	D	{1,2,3,6,f}	F	{1,2,4,5,6,f}	D
{1,2,4,6,f}	E	{1,2,3,6,f}	F	{1,2,4,5,6,f}	D
{1,2,3,6,f}	F	{1,2,3,5,6,f}	C	{1,2,4,6,f}	E

等价的DFA

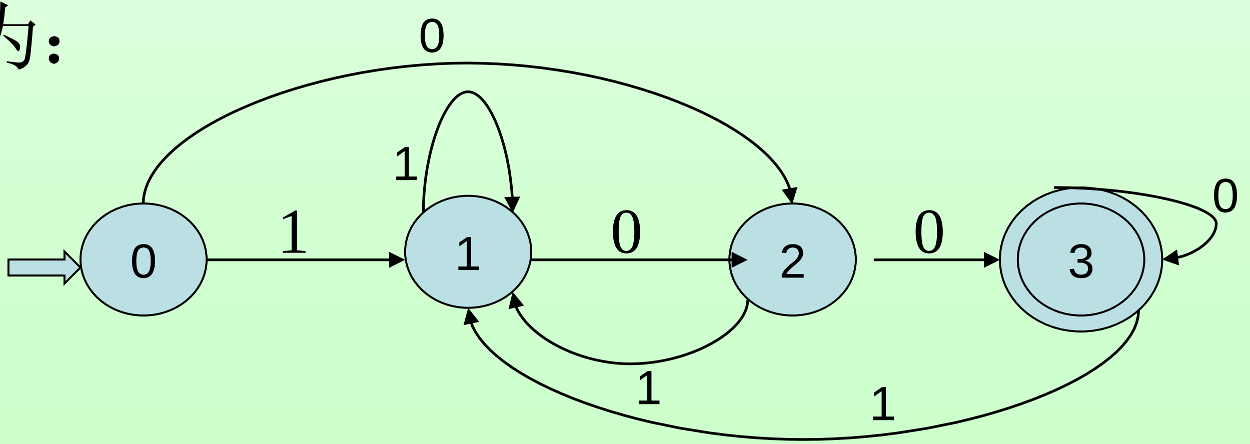


练 习

- 求下述NFA对应的DFA(完成第二步，确定化)



等价的DFA为：



DFA的化简

与某一NFA等价的DFA不一定唯一。

不同的DFA识别的正规集可能是相同的。

每一个正规集都可以由一个状态数最少的DFA所识别，这个DFA是唯一的（因状态名不同的同构情况除外）。

DFA的最小化

- **DFA的最小化**就是寻求状态数最少的DFA，即：
  - 它没有多余状态；（消去）
  - 它的状态中没有两个是互相等价的。（合并）
- **多余状态**是指：从开始状态出发，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。

- **状态S和T等价的条件**

◈ 一致性条件 —— 状态S和T必须同时为可接受状态或不可接受状态。

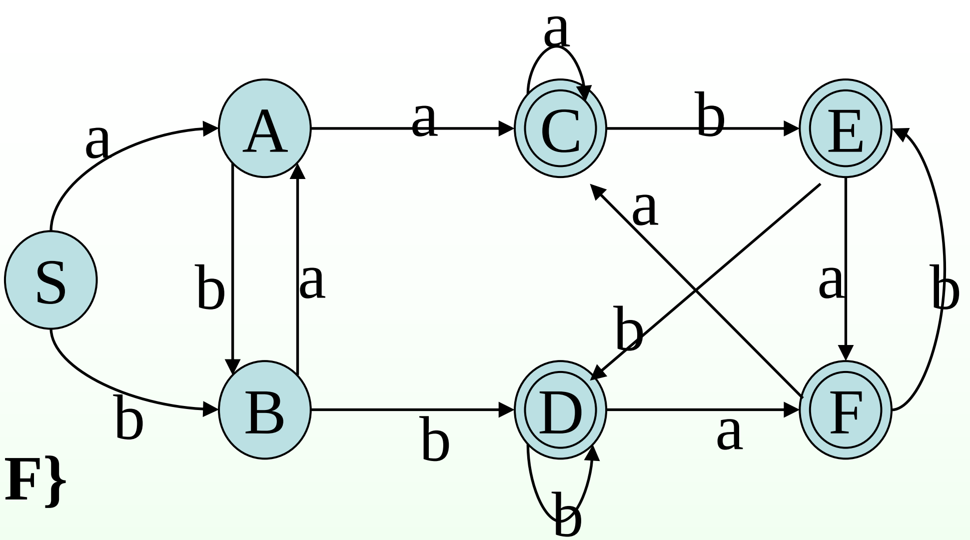
◈ 蔓延性条件 —— 对于所有输入符号，状态S和状态T必须转换到等价的状态里。

DFA的最小化的方法——分割法

- 分割法的核心
  - 把DFA的全部状态划分成一些互不相交的子集，使得任何不同的两子集的状态都是可区别的（不等价），而同一子集中的任何两个状态都是等价的.

- **算法:**
  - 所有状态分成两个子集——终态集和非终态集;
  - 运用判定状态等价的原则分别对两个子集的状态进行分析和划分，若发现某个状态与其它状态不等价，则将其作为一个新的状态子集，如果无法区分，则放在同一子集中;
  - 从每个子集中选出一个状态做代表，即可构成简化的DFA;
  - 含有原来初态的子集仍为初态，各终态的子集仍为终态。

例：化简下图的DFA



∏0: {S、A、B} {C、D、E、F}

$\downarrow$ **a**

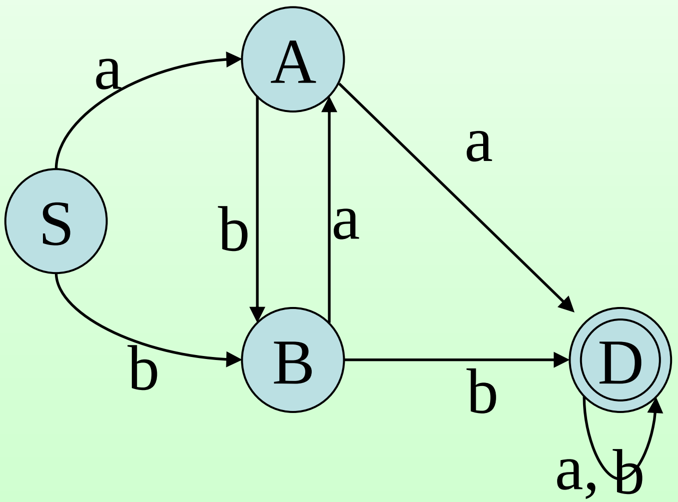
∏1: {A} {S、B} {C、D、E、F}

$\downarrow$ **b**

∏2: {A} {S} {B} {C、D、E、F}

$\downarrow$ 选C为代表

∏3: {A} {S} {B} {C}

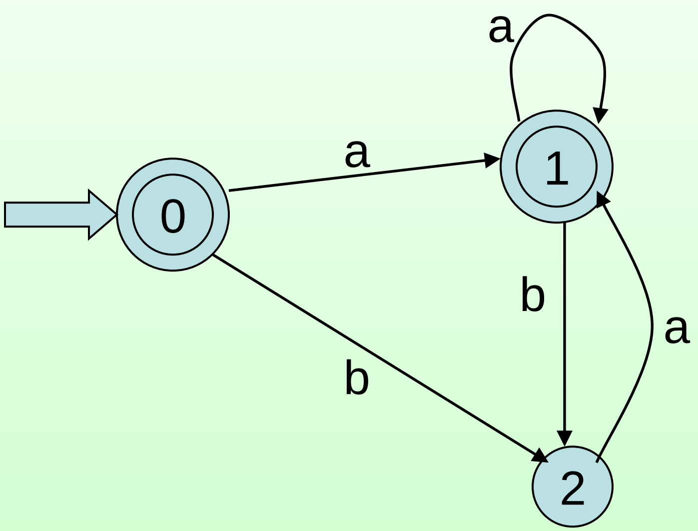


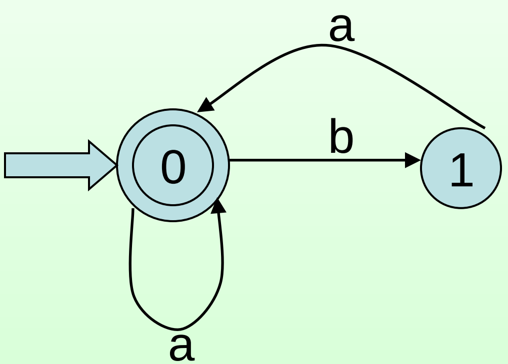
- 合并状态注意：

a、由于一个子集中，各状态等价，故只需将原进入该子集中各状态的弧都改为进入所选的状态，子集中各状态射出的弧均改为从该状态射出。
b、含有原来初态的子集仍为初态，含原终态的子集仍为终态

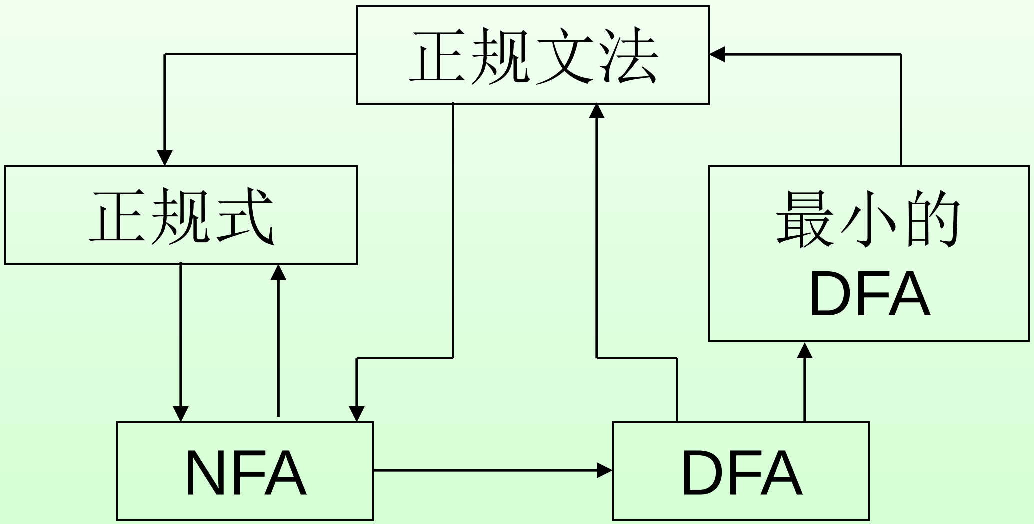
练 习

- 最小化下述DFA





- 正规集的各种描述工具及其相互间的转换



正规文法与有穷自动机的等价性

- 定义：如果L(G)=L(M), 则**正规文法G与有穷自动机M的等价。**
- 结论：
  - 对每一个右(左)线性正规文法G，都存在一个有穷自动机，使L(M)=L(G)
  - 对每一个有穷自动机，都存在一个右(左)线性正规文法G，使L(G)=L(M)

• 正规文法→有穷自动机(P51)

已知正规文法$G=(V_N, V_T, P, S)$，求相应的FA为$M=(Q, V_T, \delta, S, F)$：

1. 输入字母表：文法的终结符号V_T

2. 初始状态：就是开始符号S

3. 状态集合：增设一个终态T，以$Q=T\cup V_N$为状态结点

4. 终态集合：若P中含有$S\rightarrow\varepsilon$的产生式，则F={T, S}，否则F={T}

5. δ的计算方法(右线性文法)

(1)对P中的产生式$A\rightarrow aB$，$\delta(A, a)=B$，画从A到B的弧，标为a；

(2)对P中的产生式$A\rightarrow a$，$\delta(A, a)=T$，画从A到T的弧，标为a；

(3)对于V_T中的每个a，$\delta(T, a)=\Phi$，即在终态下无动作。

6. δ的计算方法(左线性文法)

(1)对P中的产生式$A\rightarrow Ba$，$\delta(B, a)=A$，画从B到A的弧，标为a；

(2)对P中的产生式$A\rightarrow a$，$\delta(R, a)=A$，其中R是新增的起始状态，画从R到A的弧，标为a。

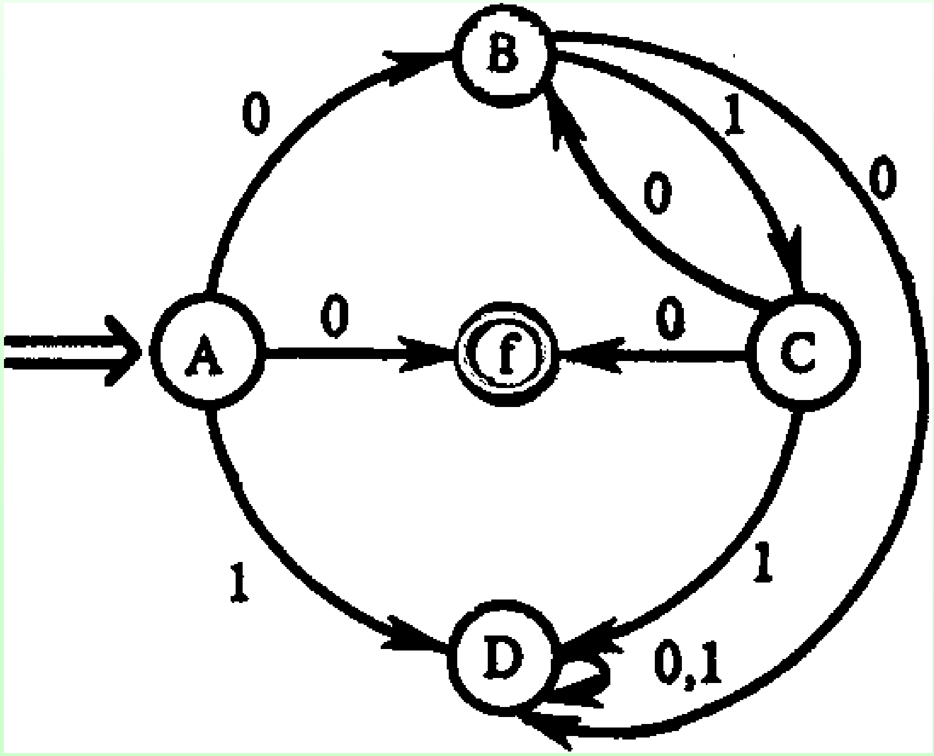
例:

$G_R = <\{0,1\},\{A,B,C,D\},A,\mathscr{P}>$,

其中产生式$\mathscr{P}$:

$A \to 0 \mid 0B \mid 1D$　　$B \to 0D \mid 1C$

$C \to 0 \mid 0B \mid 1D$　　$D \to 0D \mid 1D$



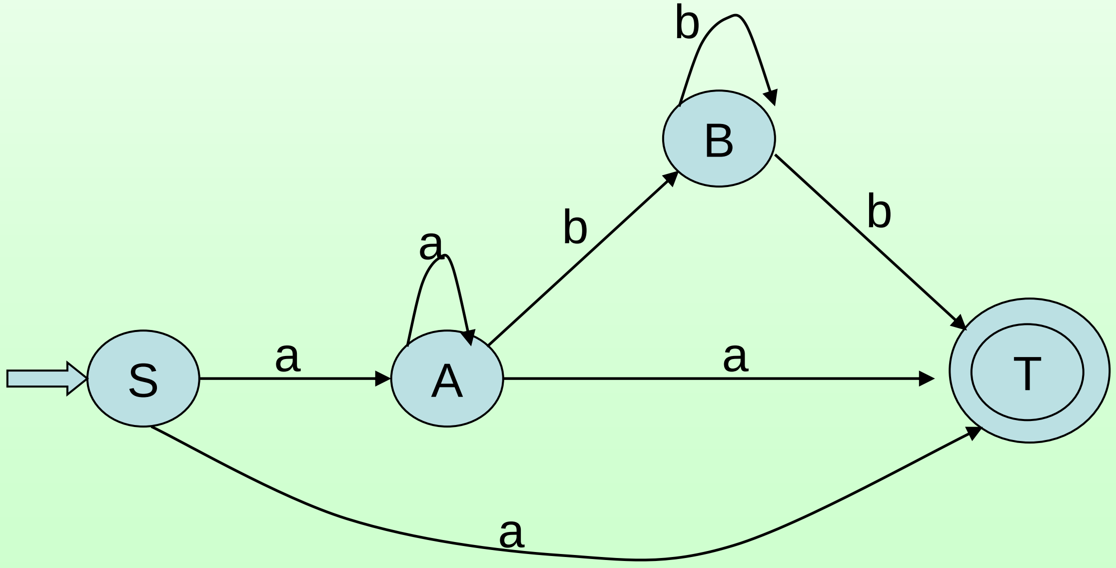
练　习

- 已知正规文法如下：
  求对应的有穷自动机

S→aA | a

A→aA | bB | a

B→ bB | b



- 有穷自动机➔正规文法

已知DFA为$M=(S,\Sigma,\delta,S_0,F)$，求相应的正规文法(右线性) $G=(\Sigma,S,S_0,P)$的方法：

1. 终结符号：V_T=字母表Σ
2. 开始符号：S=初始状态S_0
3. 非终结符号：$V_N = S$
4. 产生式：

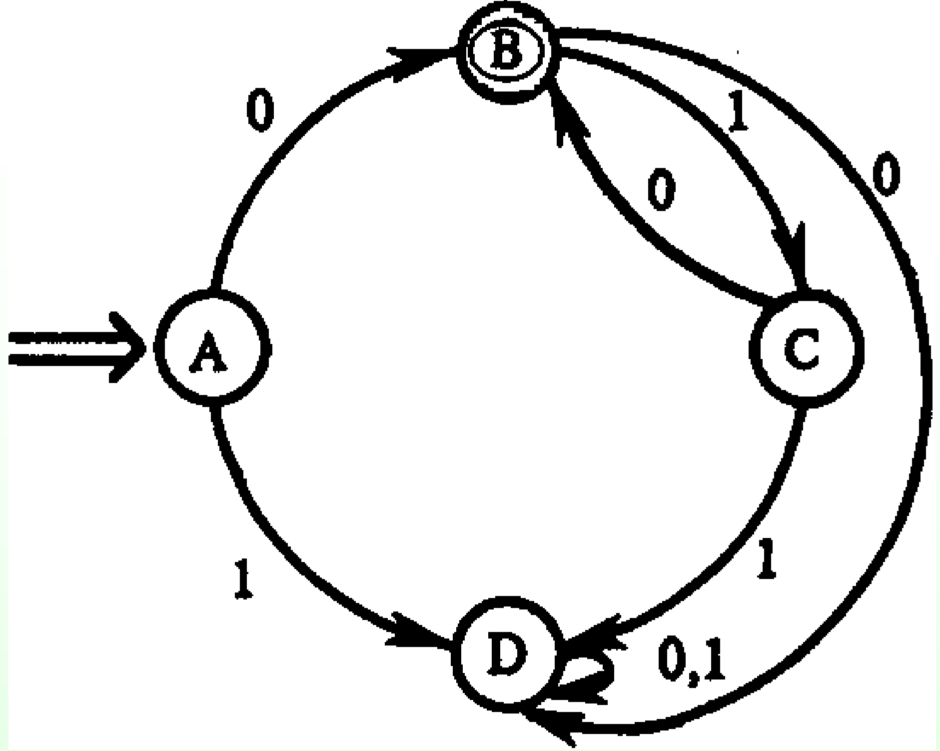
   对任何$a\in\Sigma$，$A,B\in S$，若有：$\delta(A,a)=B$，则：

   当$B\notin F$，令$A\rightarrow aB$；

   当 $B\in F$， $A\rightarrow a|aB$；

   若$S_0\in F$，增加$S_0\rightarrow\varepsilon$

例:有穷自动机为:



$G_R = <\{0,1\},\{A,B,C,D\},A,\mathscr{P}>$,

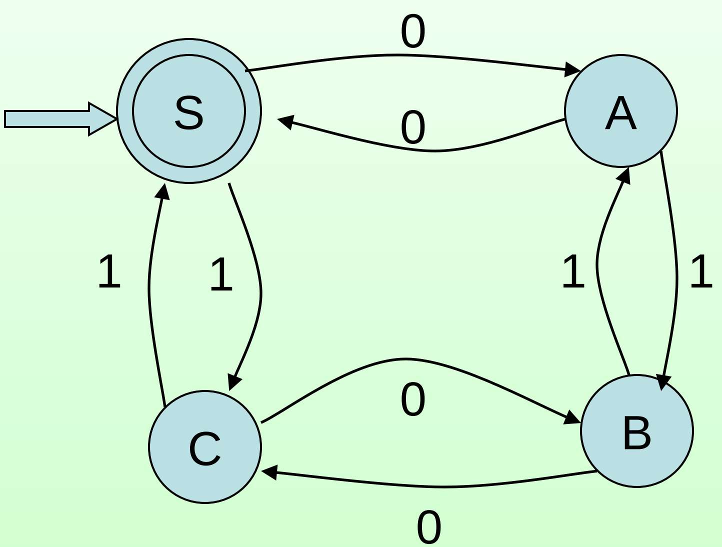
其中产生式$\mathscr{P}$:

$A \rightarrow 0|0B|1D$ $\quad$ $B \rightarrow 0D|1C$

$C \rightarrow 0|0B|1D$ $\quad$ $D \rightarrow 0D|1D$

练习

- 给出定义下述自动机的正规文法



S → 0A | 1C| ε

A → 0 | 0S | 1B

B → 1A | 0C

C → 1 | 1S | 0B

正规式与有限自动机的等价性

正规式和有穷自动机的等价性由以下两点说明：

※对于Σ上的NFA M，可以构造一个Σ上的正规式R，使得L(R)=L(M)。

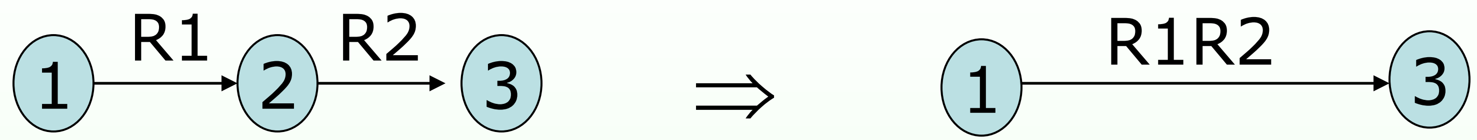
※对于Σ上的每个正规式R，可以构造一个Σ上的NFA M，使得L(M)=L(R)。

- 有穷自动机M➔正规式R

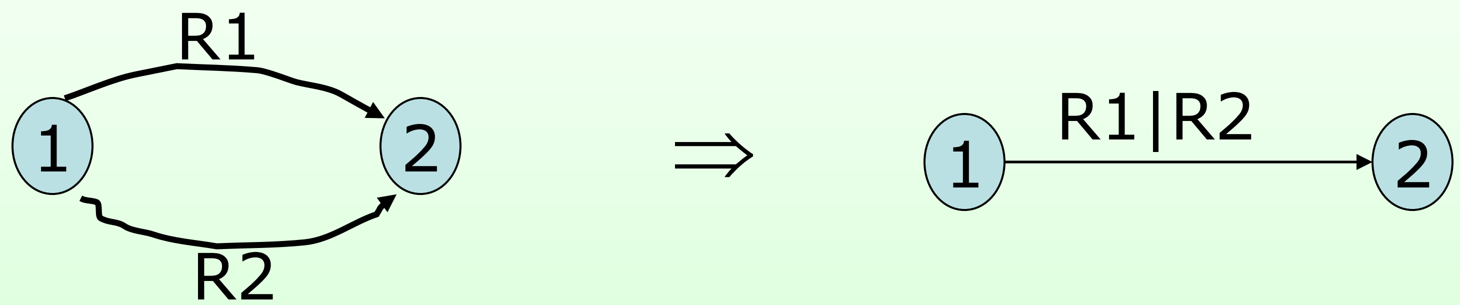
方法：

- 将状态转换图的概念拓广，每一条弧用一个正规式作标记。
- 增加结点X,Y,使之成为新的唯一的初态和终态。从X引 **ε** 弧到原初态结点, 从原终态结点引 **ε** 弧到Y结点。
- 利用如下的规则消去新的状态图中的所有结点，直至只剩下X,Y结点。
- 最后得到从X到Y弧上的正规式。

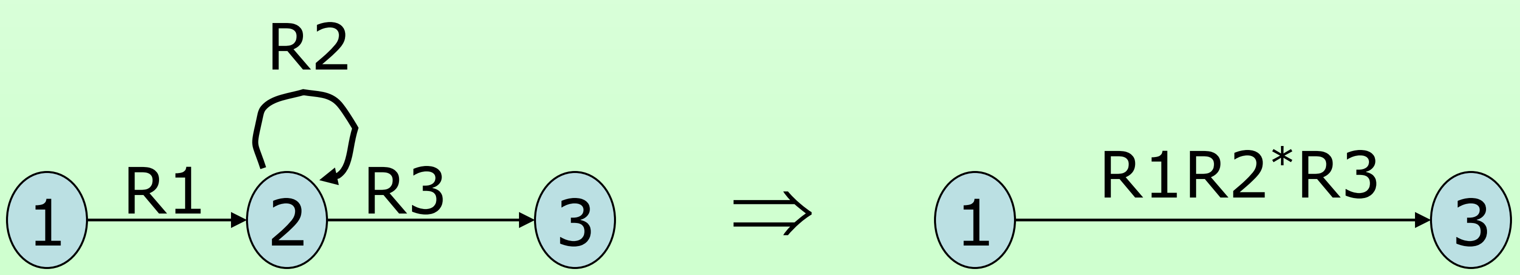
a.



b.

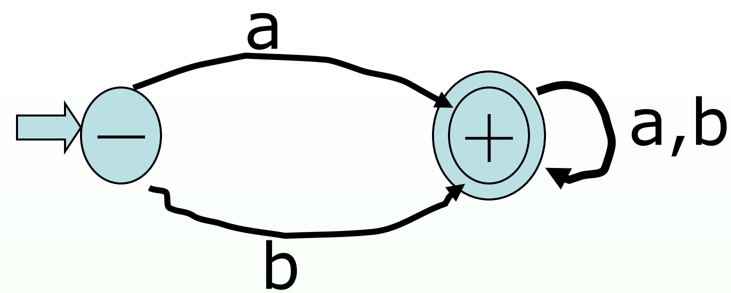


c.

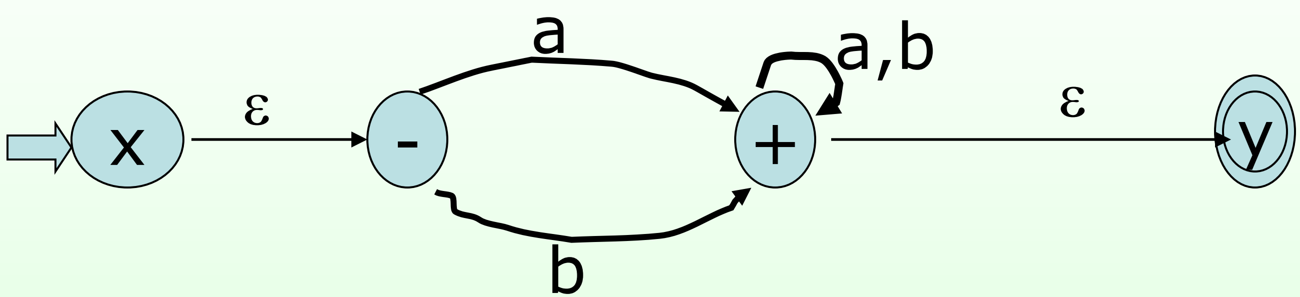


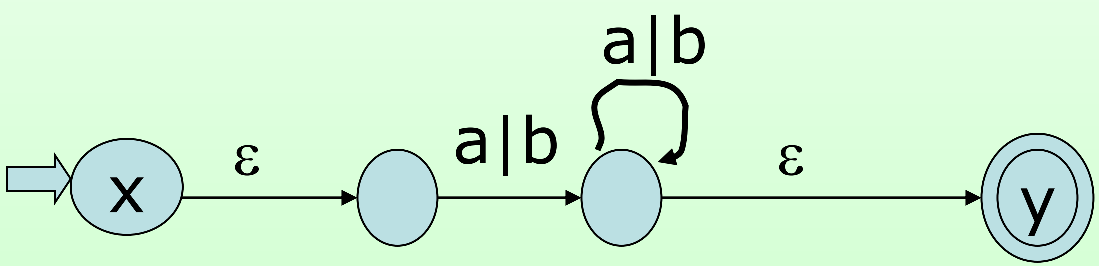
例: L(M)如右图:

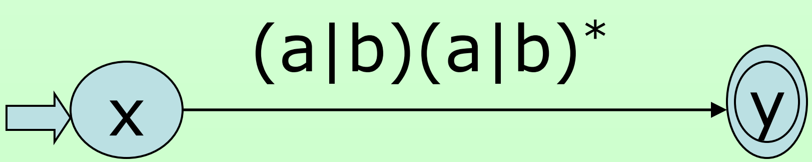
求正规式R，使L(R)=L(M).



解:





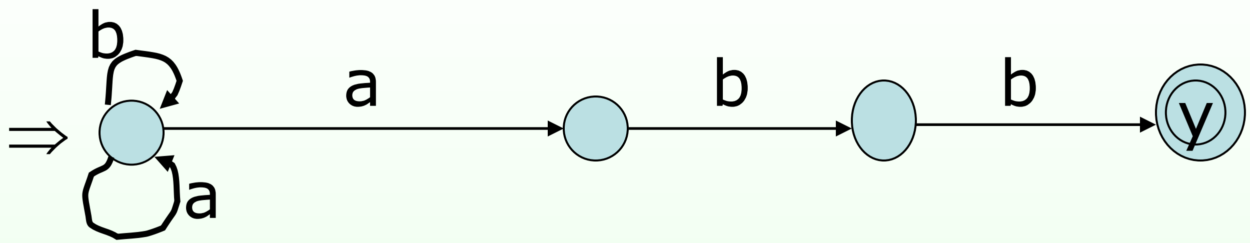


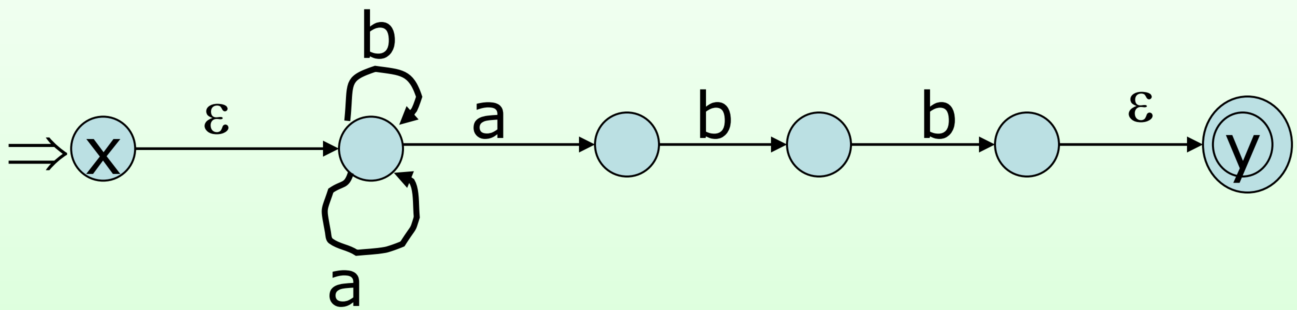
因此:

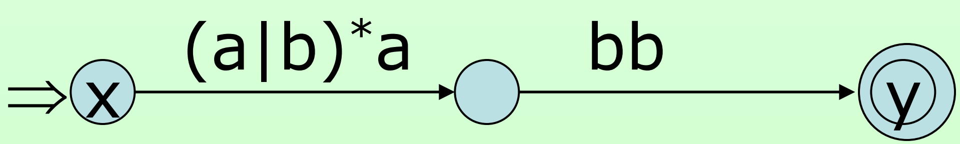
L(R)= (a|b)(a|b)*

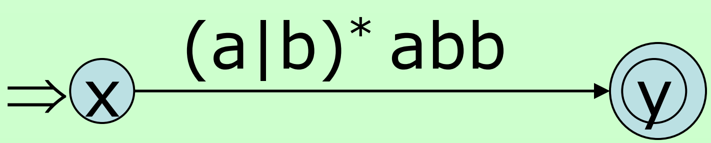
练 习

求下述有穷自动机对应的正规式







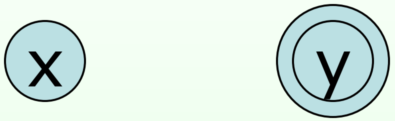


$L(R) = (a|b)^*abb$

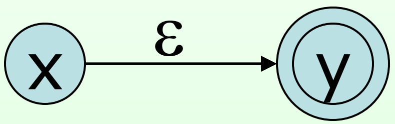
• 正规式R➔有穷自动机NFA M(P54)

方法如下：

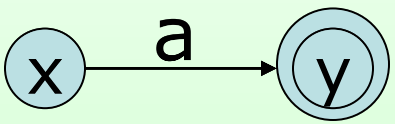
① 正规式φ，构造NFA为：⇒



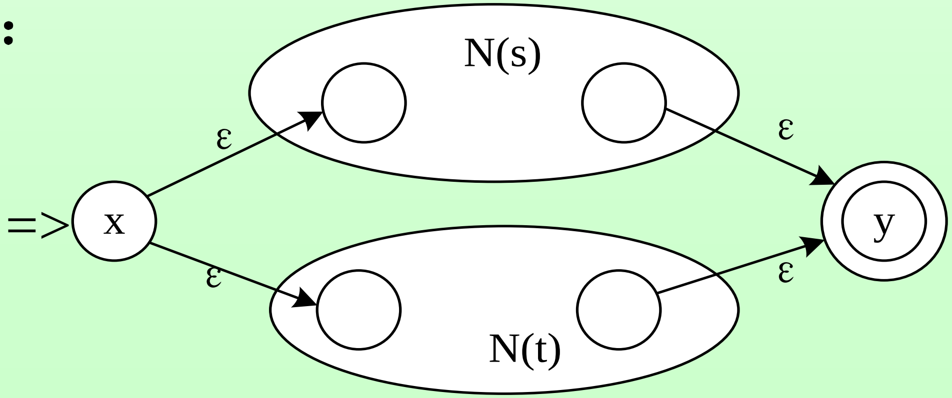
② 对应正规式ε，构造NFA为： ⇒



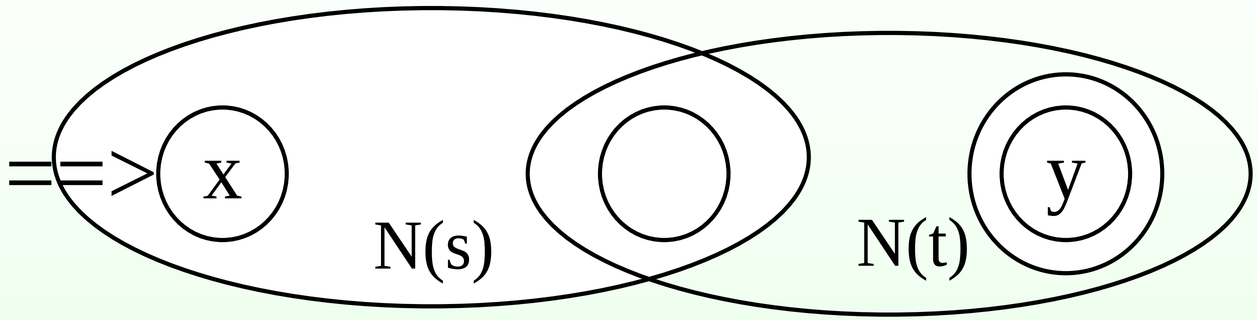
③ 对应正规式a，构造NFA为： ⇒



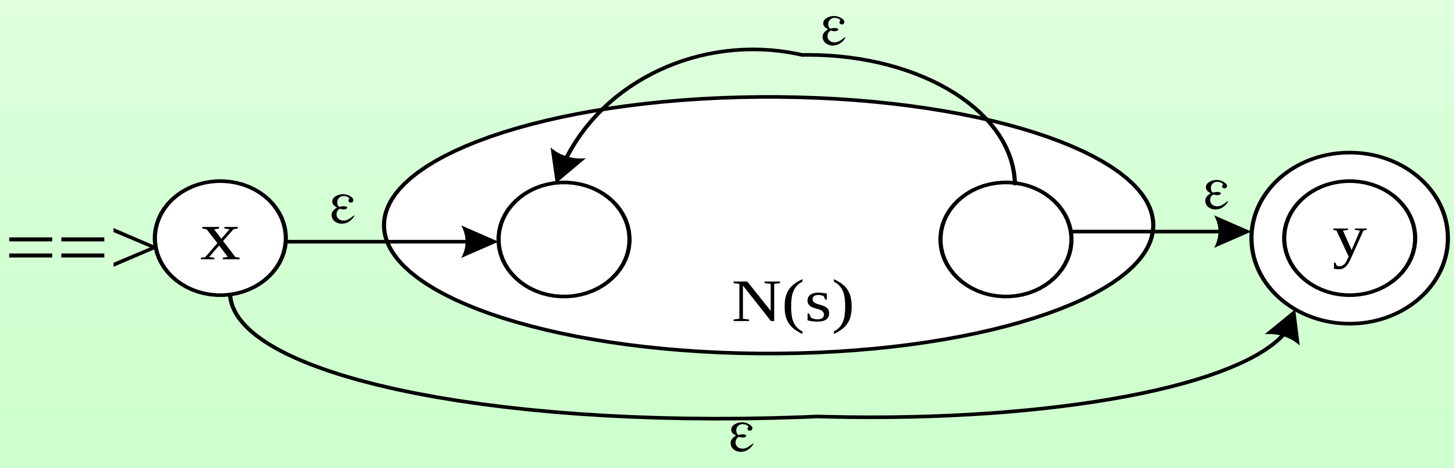
④ s,t是正规式，相应NFA为N(s),N(t)，则正规式R=s|t，构造NFA(R) 为：



⑤ s,t是正规式，相应NFA为N(s),N(t)，则正规式R=st，构造NFA(R) 为：

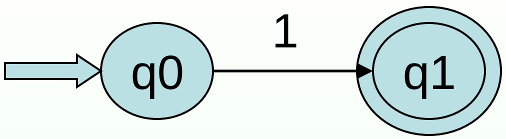


⑥ s是正规式，相应NFA为N(s) ，则正规式R=s*，构造NFA(R) 为：

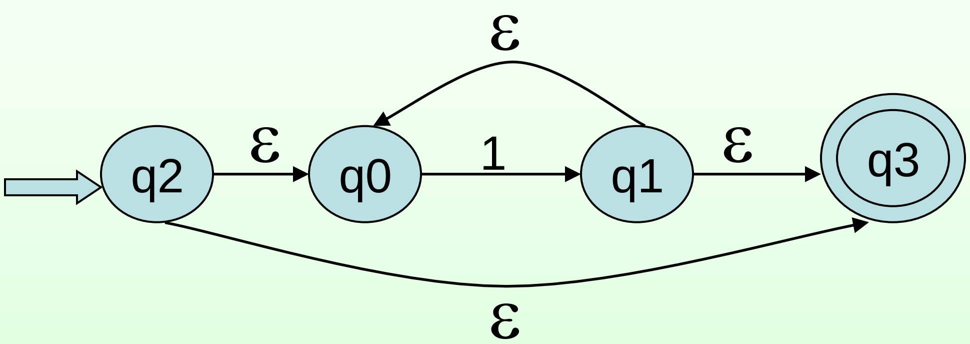


书上例3.5　　P56

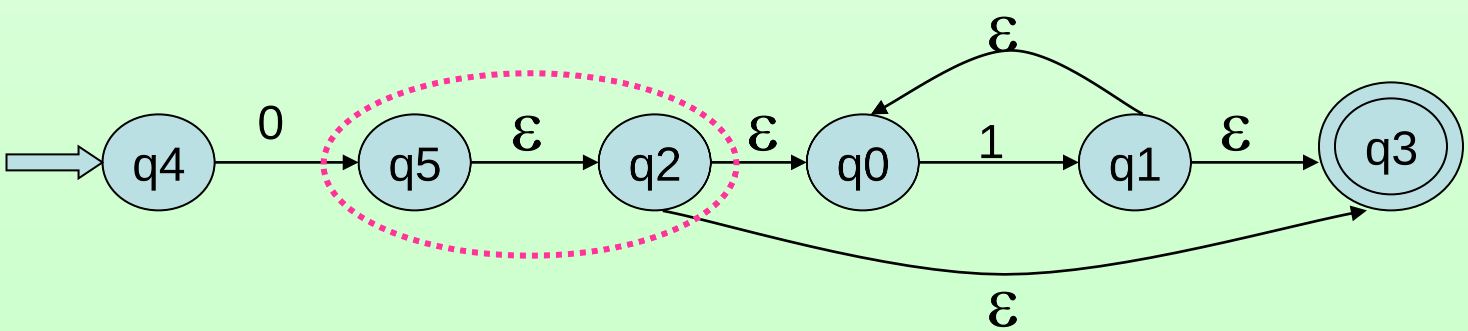
R = 1的有穷自动机:



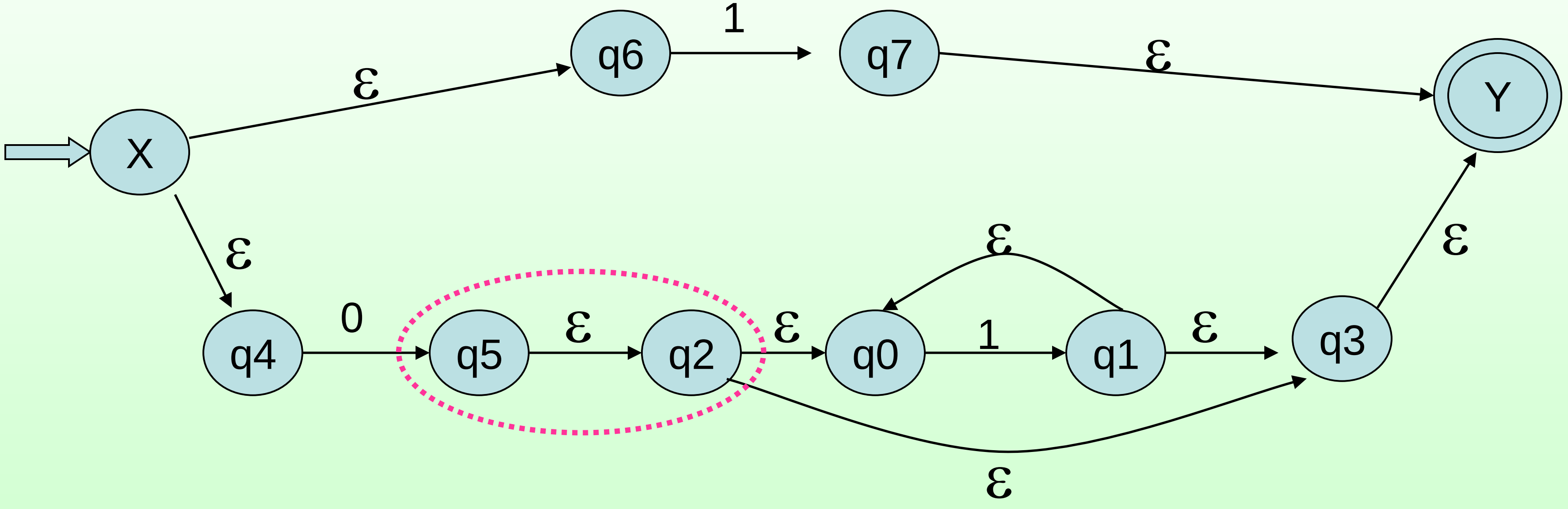
R = 1*的有穷自动机:



R = 01*的有穷自动机:

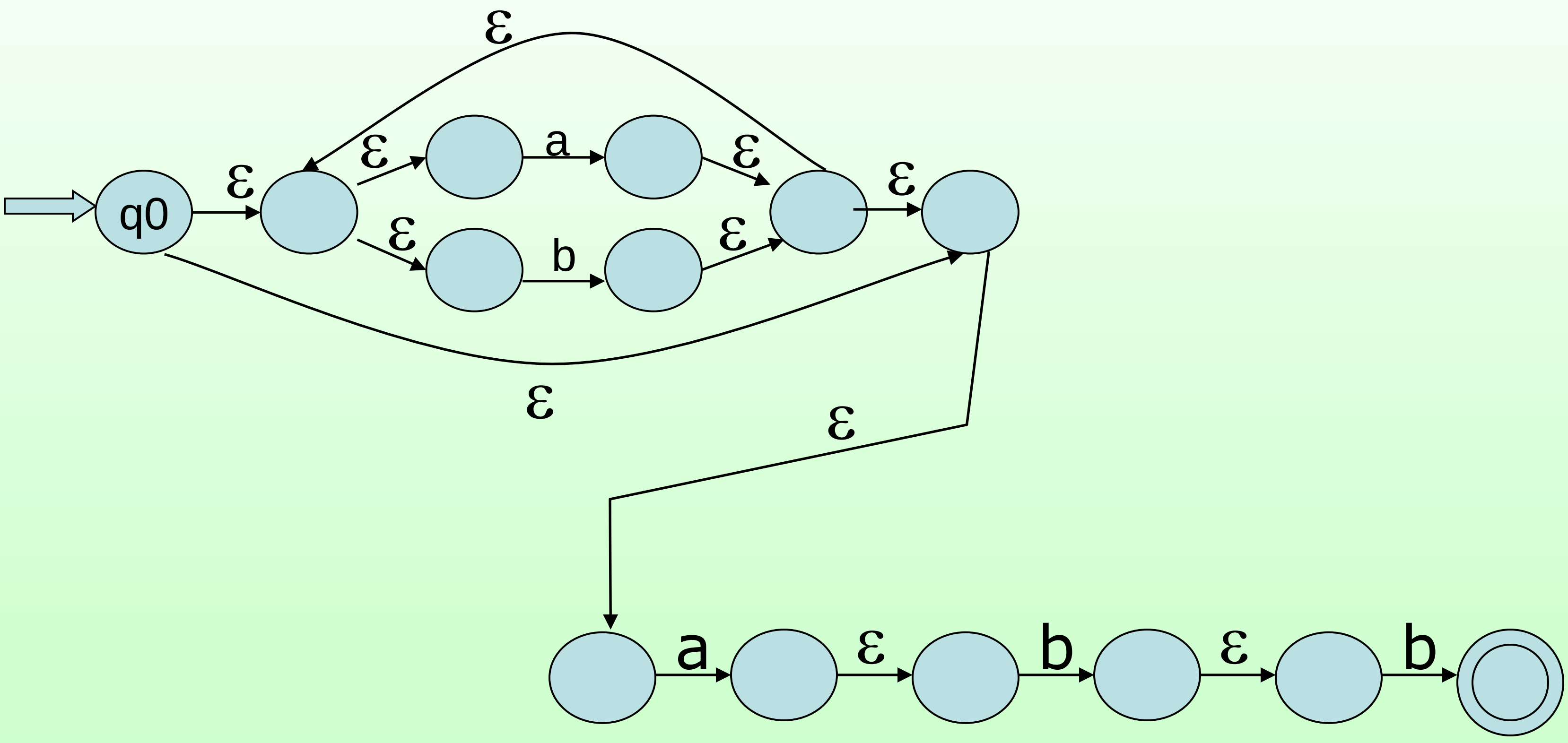


R = 01* | 1的有穷自动机:



练习

正规式L(R) =(a|b)*abb，构造NFA使L(N)=L(R)



本章小结

- 本章要求
  - 1．词法分析器的作用和接口，用高级语言编写词法分析器等内容，它们与词法分析器的实现有关。
  - 2．掌握下面涉及的一些概念，它们之间转换的技巧、方法或算法。
    - NFA $\rightarrow$ DFA
    - DFA $\rightarrow$ 最简DFA
    - 非形式描述的语言 $\leftrightarrow$ 正规式
    - 正规式 $\leftrightarrow$ NFA
    - 正规文法 $\leftrightarrow$ NFA

课堂练习(1~3章)

$S \rightarrow aAb$
$A \rightarrow BcA \mid B$
$B \rightarrow idt \mid \varepsilon$

- 1. 有文法G[S]:
  问：符号串aidtcBcAb、ab、abidt是否是该文法的句型？为什么？
- 2. 编译原理各个阶段各应遵循哪些原则？
- 3. 写出不能被5整除的偶整数的正规文法和正规式
- 4. 有一台自动售货机，接受1元和5角的硬币，出售每瓶1元5角的饮料，顾客每次向机器中投放≥1元5角的硬币，就可得到一瓶饮料(注：每次只给一瓶饮料，且不找钱)，构造该售货机的有穷自动机。
- 5. 设计一个状态数最少的DFA, 其输入字母表是{0, 1}，它能接受以00或01结尾的所有序列,并给出相应的正规文法。

3.4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，使的L(M)=L(R)。

3.4.1 Lex的概述