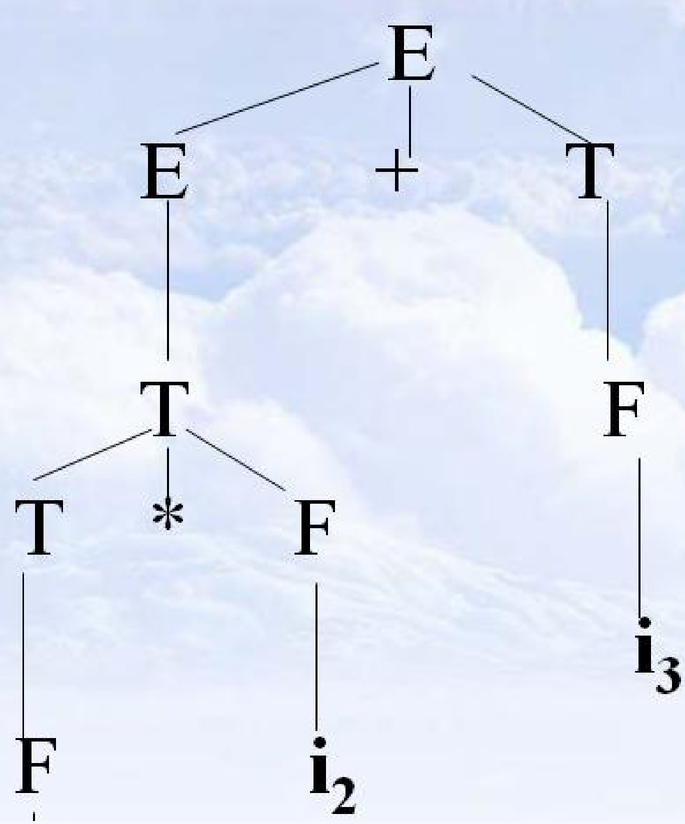
复 习

- 文法的直观概念
- 符号和符号串
- 文法和语言的形式定义
- 文法的类型(0、1、2、3)
- 上下文无关文法及其语法树（最左/右推导、二义性）
- 句型的分析（自上而下/自下而上、短语、直接短语、句柄）
- 有关文法实用中的一些说明



G[E]： $E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid i$

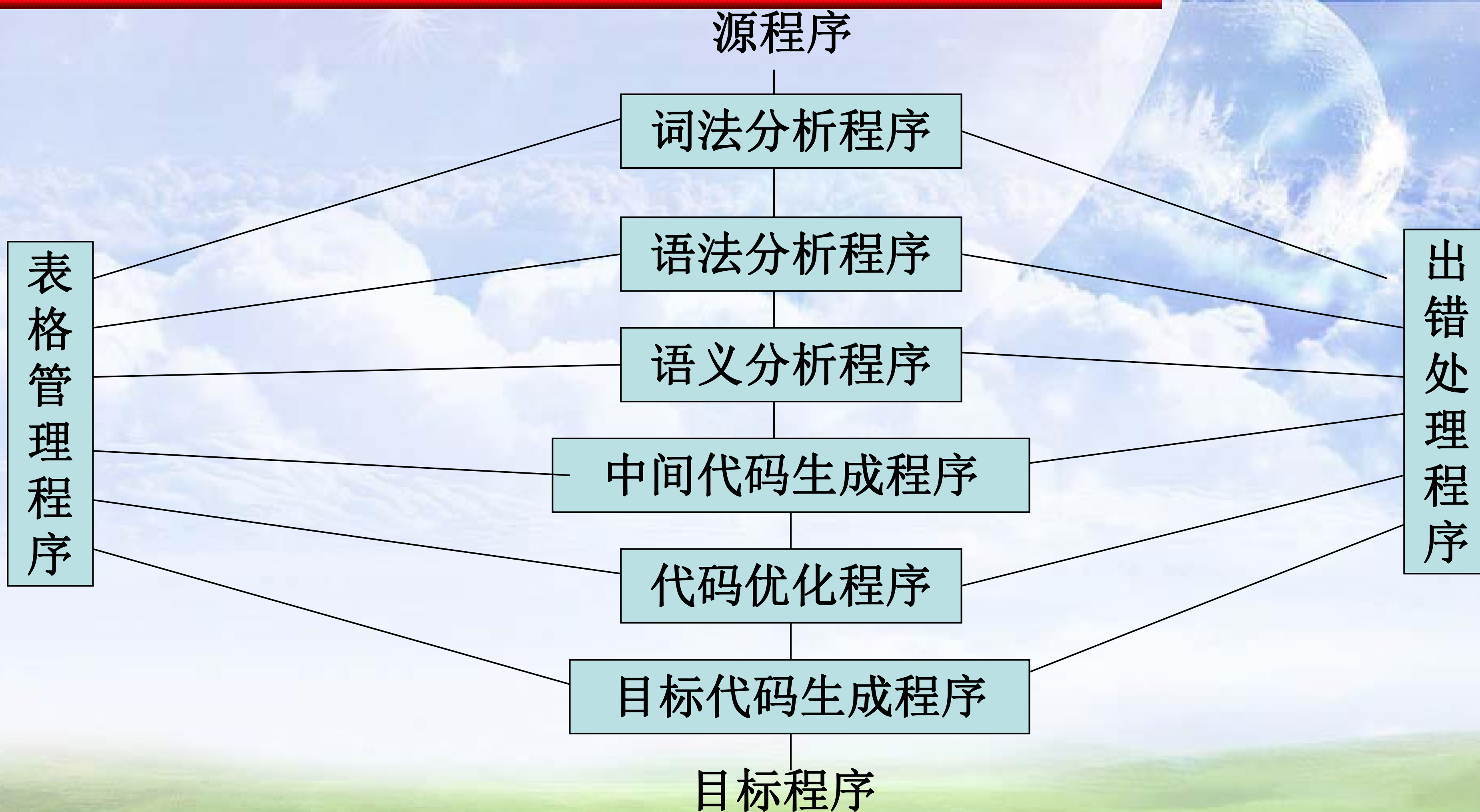
句型：$F*i+i$

短语：$F*i_2+i_3$， $F*i_2$， F， i_2， i_3

直接短语：F， i_2， i_3

句柄：F

编译程序的结构


源程序
词法分析程序
语法分析程序
语义分析程序
中间代码生成程序
代码优化程序
目标代码生成程序
目标程序
表格管理程序
出错处理程序

第三章　词法分析

主要内容

讨论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。

3.1 词法分析程序
3.2 单词的描述工具
3.3 有穷自动机
3.4 正规式和有穷自动机的等价性
3.5 正规文法和有穷自动机的等价性
3.6 词法分析程序的自动构造

3.1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。**

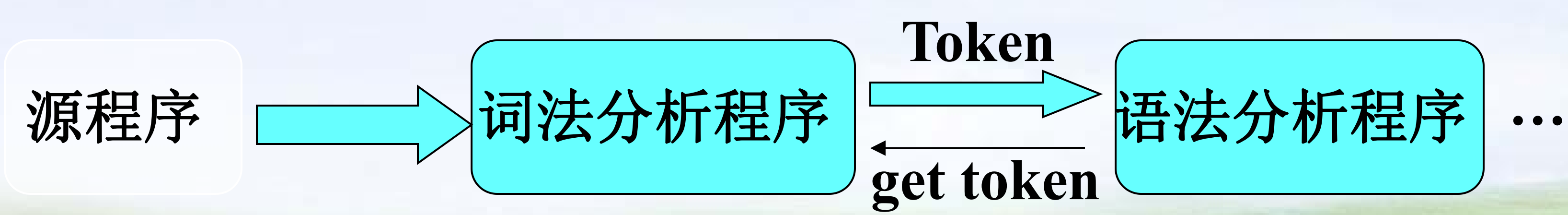
词法分析的任务

- 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，……

词法分析程序与语法分析程序的接口方式

方式一：词法分析程序作为**独立的一遍**，把字符流的源程序变为单词序列，输出在一个中间文件上，这个文件作为语法分析程序的输入继续编译过程。

方式二：把词法分析程序**设计成一个子程序**，每当语法分析程序需要一个单词时，则调用该子程序。词法分析程序每得到一次调用，便从源程序文件读入一些字符，直到识别出一个单词，或说直到下一个单词的第一个字符为止。



词法分析的输出

- **词法分析的功能是读入源程序，输出单词符号。**
- **输出的单词符号一般分成5种：**
  1. **关键字(基本字):** begin end for do if else……
  2. **标识符:** 由用户定义，表示各种名字
  3. **常数:** 整常数、实常数、布尔常数、字符串常数等
  4. **运算符:** 算术运算符+、-、*、/等；逻辑运算符not、or与and等；关系运算符=、<>、>=、<=、>和<等
  5. **界符:** ，、；、（、）...
- **词法分析程序所输出的单词符号采用二元式表示：**

(单词种别, *单词自身的值*)

表示单词的种类，可用整数编码或记忆符表示

不同的单词不同的值

词法分析的输出举例

设标识符编码为1，
常数为2，
关键字为3，
运算符为4，
界符为5

程序段:

if i=5
then x:=y;

输出

(3, 'if')
(1, 指向i的符号表入口)
(4, '=')
(2, '5')
(3, 'then')
(1, 指向x的符号表入口)
(4, ':=')
(1, 指向y的符号表入口)
(5, ';')

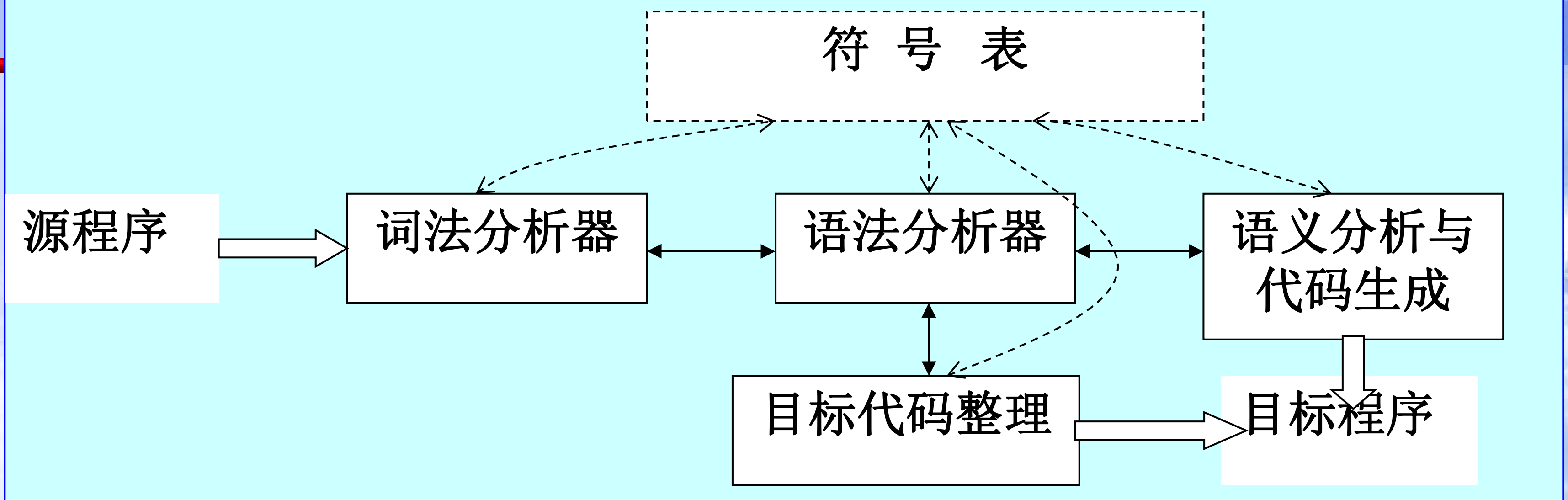
词法分析阶段的错误处理

1. 非法字符检查
2. 关键字拼写错误检查
3. 不封闭错误检查
4. 重复说明检查
5. 错误恢复与续编译

紧急方式恢复(panic-mode recovery)

反复删掉剩余输入(源程序的)最前面的字符，直到词法分析器能发现一个正确的单词为止。

词法分析器的位置



图（以语法分析器为中心）

- 以语法分析器为中心的优点：
  - 简化编译器的设计。
  - 提高编译器的效率。
  - 增强编译器的可移植性。

单词的描述工具

程序设计语言中的单词是基本语法符号。单词符号的语法可以用有效的工具加以描述，并且基于这类描述工具，可以建立词法分析技术，进而可以建立词法分析的自动构造方法。

1. 正规文法
2. 正规式
3. 有穷自动机

1. 正规文法

回顾一下3型文法，设文法G=(V_N, V_T, P, S), 若P中每一个产生式都是A→aB或A→a，其中A, B∈V_N，a∈V_T^*，则G是一个3型文法或正规文法。

各类单词的正规文法描述

用l表示a~z中的任一个英文字母，d表示0~9中的任一个数字。

标识符的文法：

<标识符> →l | l<字母数字>

<字母数字> →l | d | l<字母数字> | d<字母数字>

无符号整数的文法：

<无符号整数> →d | d<无符号整数>

运算符的文法：

<运算符> →+ | - | * | / | = | <<等号> | ><等号>……

其他文法：

<等号> →=

<界符> →, | ; | (|) | ……

各类单词的正规文法描述

无符号实数的文法：

〈无符号实数〉→ d〈余留无符号数〉| .〈十进小数〉| e〈指数部分〉

〈余留无符号数〉→ d〈余留无符号数〉| .〈十进小数〉| e〈指数部分〉| ε

〈十进小数〉 → d 〈余留十进小数〉

〈余留十进小数〉 → e〈指数部分〉| d 〈余留十进小数〉| ε

〈指数部分〉 → d 〈余留整指数〉| s〈整指数〉

〈整指数〉 → d 〈余留整指数〉

〈余留整指数〉 → d 〈余留整指数〉 | ε

其中s表示正或负号，d表示0~9中的任何一个数字。

如 25.55e+5 、120 和 2.1

2. 正规式

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

下面是正规式和它所表示的正规集的递归定义。

正规式的定义

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Σ`={Φ，ε，|，•，*，(，)}。

1. ε和Φ都是Σ上的正规式，所表示的正规集分别为{ε}和{ }；
2. 任何a∈ Σ，a是Σ上的一个正规式，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，所表示的正规集分别为$L(e_1)$和$L(e_2)$，那么，(e_1), $e_1 | e_2$, $e_1•e_2$, 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

- "｜"读为"或"（也有使用"+"代替"｜"的）；
- "• "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。

正规式的代数规律

- 设r,s,t为正规式，正规式服从的代数规律有：

r｜s=s｜r	"或"服从交换律
r｜(s｜t)=(r｜s)｜t	"或"的可结合律
(rs)t=r(st)	"连接"的可结合律
r(s｜t)=rs｜rt (s｜t)r=sr｜tr	分配律
εr=r, rε=r	ε是"连接"的恒等元素 零一律
r｜r=r r*=ε｜r｜rr｜…	"或"的抽取律

例子

令Σ={a，b}，Σ上的正规式和相应的正规集的例子有：

正规式	正规集
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}

正规式举例

例3.1 :令Σ={l，d}，Σ上的正规式 r=l(l｜d) *

定义的正规集为: {l,ll,ld,ldd,……},

- 其中l代表字母,d代表数字,正规式 即是 字母(字母|数字) *
- 表示的正规集中每个元素的模式是“字母打头的字母数字串”
- 是Pascal和 多数程序设计语言允许的的标识符的词法规则.

例3.2: Σ={d，•，e，+，-},

Σ上的正规式 d*(•dd *｜ε)(e(+｜- ｜ε)dd* ｜ε)

- 其中d为0~9的数字。
- 表示的是无符号数的集合。

程序设计语言的单词都能用正规式来定义.

正规式的构造

正规集	正规式
{0}	0
{00}	00
{01}	01
{00,01}	0(0|1)
$\{0^n1^m \mid n>=1,m>=1\}$	00*11*
$\{0^n \| 1^n \mid n>=1\}$	$0^+ \| 1^+$

- 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： $e_1= (a|b)$， $e_2 = b|a$
  - 又如： $e_1= b(ab)^*$ ， $e_2 =(ba)^*b$
    $e_1= (a|b)^*$ ，$e_2 =(a^*|b^*)^*$

正规文法和正规式的等价性

一个正规语言可以用正规文法定义，也可以由正规式定义。

对任意一个正规文法，存在一个定义同一个语言的正规式。

对每个正规式，存在一个生成同一语言的正规文法。

正规式→正规文法

问题：将Σ的一个正规式r转换成文法(V_N, V_T, P, S)

转换方法：令V_T=Σ，确定产生式和V_N的元素用如下方法：

1. 引入开始符号S生成类似的正规产生式形式S→r。

2. 若x和y都是正规式：

– 对于A→xy的正规式产生式，

重写成A→xB，B→y，B是新选的非终结符，B∈V_N。

正规式→正规文法

- 对形如A → x | y的正规式产生式，

  重写为A →x，A →y
- **对形如A→x*** 的正规式产生式，

  重写为A →xA，A →ε
- **对形如A→x*y**的正规式产生式，

  重写为A →xA，A →y
- **对形如A→x⁺** 的正规式产生式，

  重写为A →xA，A →x

3.不断利用上述规则做变换，直到每个产生式都符合正规文法的格式

正规式→正规文法举例

将r=a(a|d)*转换成相应的正规文法。

令S是文法的开始符号，

首先形成S→a(a|d)*，

然后形成S →aA， A →(a|d)*，

再变换成S →aA， A →(a|d)A， A → ε

进而变换成符合正规文法产生式的形式

S →aA， A → aA， A→ dA， A → ε

正规式向正规文法的转换规则

正规式	正规文法
xy	A→xB B→y
x | y	A →x A →y
x*	A →xA A →ε
x*y	A →xA A →y
x^+	A →xA A →x

正规式➔正规文法举例

- 将r=(a|d)(a|d)*转换成相应的正规文法。

S →aA， S →dA A → aA， A→ dA， A → ε

- 将r=d^+(ε| . d^+)转换成相应的正规文法。

S→dA， S→dB, A→dA， A→ ε ,

B→dB， B→.C C→dD, D→dD， D→ ε (实数)

- 将r=a(a|b)*(ε| ((. | _)(a|b)(a|b)*))转换成相应的正规文法。

S→aA， S→aC, A→aA， A→bA， A→ ε ,

C→aC， C→bC C→.B, C→_B,

B→aA， B→bA B→a, B→b

- 将r=0(0|1)$^+$转换成相应的正规文法。

S →0A， A→0A A → 1A， A→0， A →1

整数：d^+ 科学计数法: $d^+.d^+Ed^+$

复习

- 词法分析程序
- 单词的描述工具
- 正规文法
- 正规式：递归定义、运算符优先级、正规式的构造
- 正规式和正规文法的等价性：相互转化

正规文法→正规式

基本上是正规式→正规文法的逆过程，最后只剩下一个开始符号定义的正规式，转换规则如下：

	文法产生式	正规式
规则1	A →xB，B →y (必须包含A和B的所有产生式)	A=xy
规则2	A →xA | y (必须包含A的所有产生式)	A=x*y
规则3	A →x，A →y	A=x|y

正规文法➔正规式举例

例：文法G[S]: S→aA, S→a, A→aA, A→dA, A →a, A →d转换为正规式

首先有：**S=aA|a　　A=(aA|dA)|(a|d)**

将A的正规式变换为**A=(a|d)A|(a|d)，**

又变换为：**A=(a|d)*(a|d)，**

再将A的右端代入S的正规式得：**S=a(a|d)*(a|d)|a**

再利用正规式的代数变换可依次得到

S=a((a|d)*(a|d)| ε)　S=a(a|d)*

即a(a|d)*为所求。

正规文法→正规式举例

例：将下面的文法转换为正规式

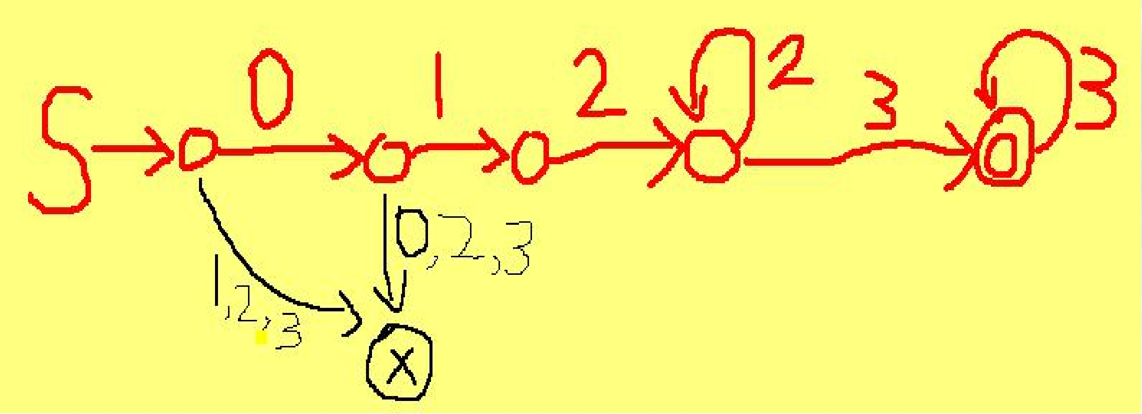
S→0A　A→1B　B→2B|2C　C→3C|3

S→01B

$S \to 012^*2C$

$S \to 012^*23^*3$

$S \to 012^+3^+$



研究0122333的识别过程

有穷自动机

- 一个有穷状态集；
- 一个输入符号表，它的每个元素称为一个输入符号；
- f是转换函数：

  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；
- 一个初态；
- 一个终态集，终态也称可接受状态或结束状态。

3.3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) 。

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

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

- 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是转换函数，是在K×Σ→K上的映射，如f（k_i，a）=k_j，（k_i，k_j∈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；
- S∈K是唯一的一个初态；
- 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

状态图

- 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 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 的例子：

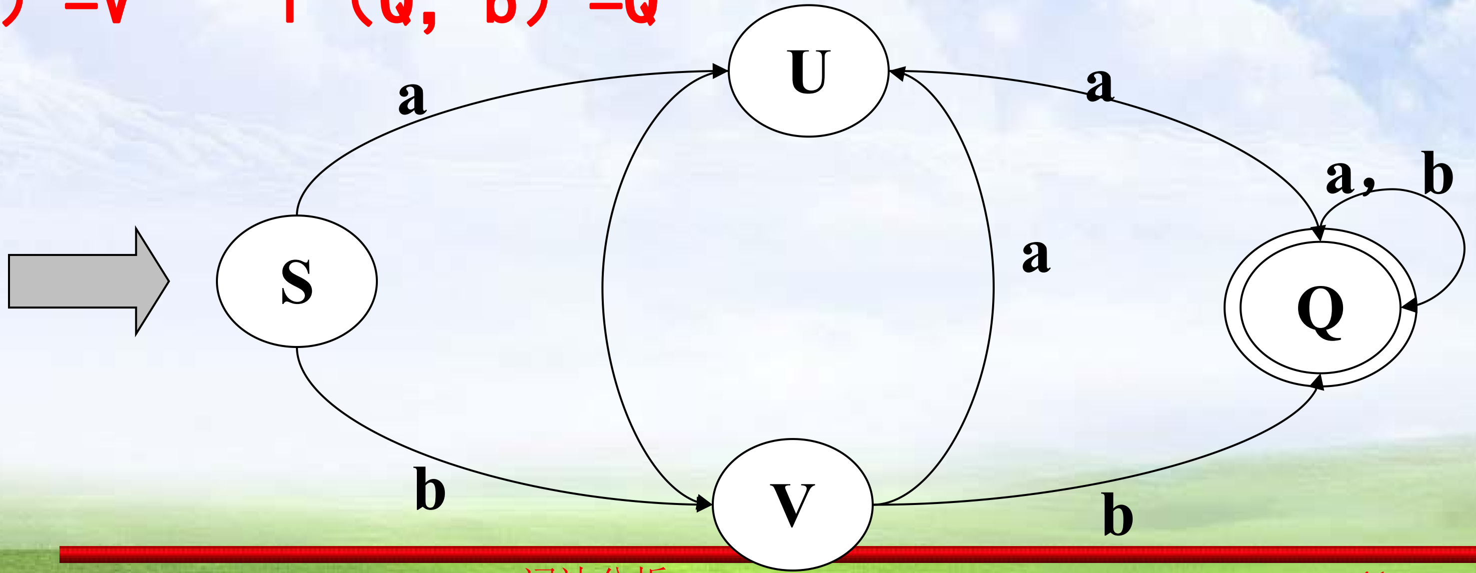
DFA M=（{S，U，V，Q}，{a，b}，f，S，{Q}）

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还可以用一个矩阵表示

- 该矩阵的一行表示一状态，一列表示一个输入字符，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　M=（{S，U，V，Q}，{a，b}，f，S，{Q}）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的矩阵表示

DFA　M=（{S，U，V，Q}，{a，b}，f，S，{Q}）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

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

运行

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t_1t_x的形式，其中$t_1 \in \Sigma$，$t_x \in \Sigma^*$）在DFA M=（K，Σ，f，S，Z）上运行的定义为：

f（Q，t_1t_x）=f（f（Q，t_1），t_x）　其中Q∈K扩充转换函数f为 K×Σ*→K上的映射，且：f（k_i，ε）= k_i

接受

$\sum^*$上的符号串t被DFA M接受

M=（K，Σ，f，S，Z）

若t∈ $\sum^*$，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）。

即，若存在一条从初态结点到某一终态结点的道路，且这条道路上所有弧的标记符连接成的符号串等于t，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）

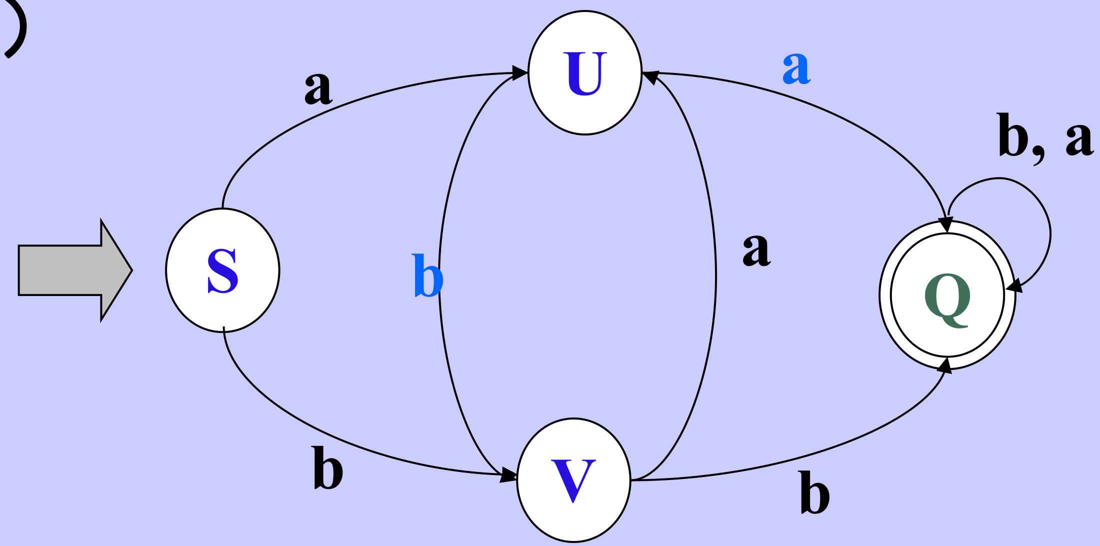
=f（U，ab）

=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

- **DFA的确定性表现在转换函数f:K×Σ→K是一个单值函数，也就是说，对任何状态k∈K，和输入符号a∈Σ，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。**

DFA的行为的程序模拟.

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’)**

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是一个非空的初始状态集，Z ⊂K为终止状态集.

NFA举例

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}

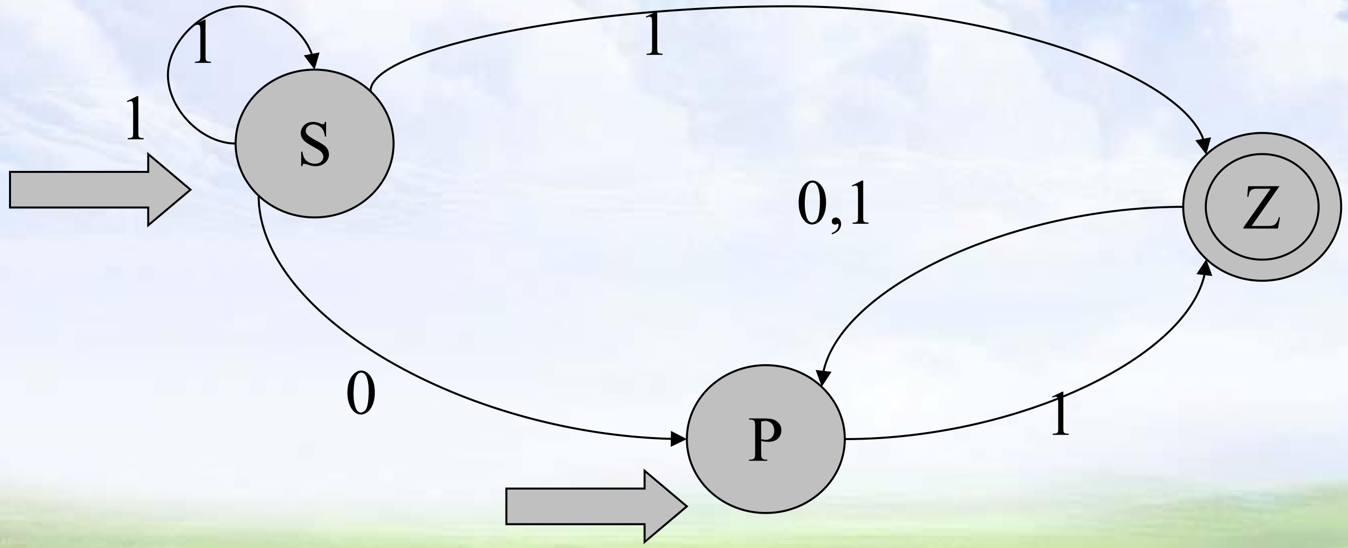
f（S，1）={S，Z}

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



矩阵表示

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}

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

类似DFA, 对NFA M=｛K，∑，f，S，Z｝也有如下定义

- ∑*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t_1t_x的形式，其中$t_1 \in \sum$，$t_x \in \sum^*$）在NFA M上运行的定义为：
  $f(Q, t_1t_x) = f(f(Q, t_1), t_x)$ 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，那么空串可为M所接受。

000

111

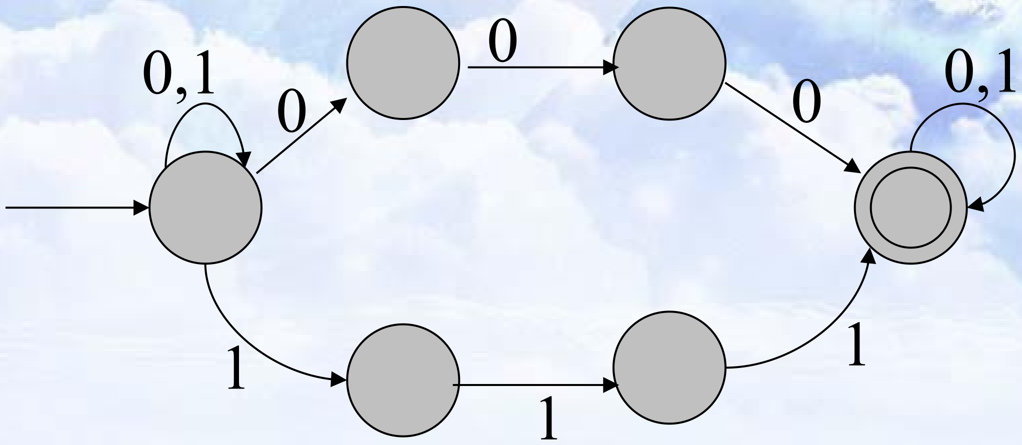
1010001

110000001

10001011

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$V \subset \Sigma^*$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)。

DFA和NFA的关系

- 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，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确定化基本思路

- 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使用它的一个状态去记录在NFA读入一个输入符号后可能达到的所有状态.

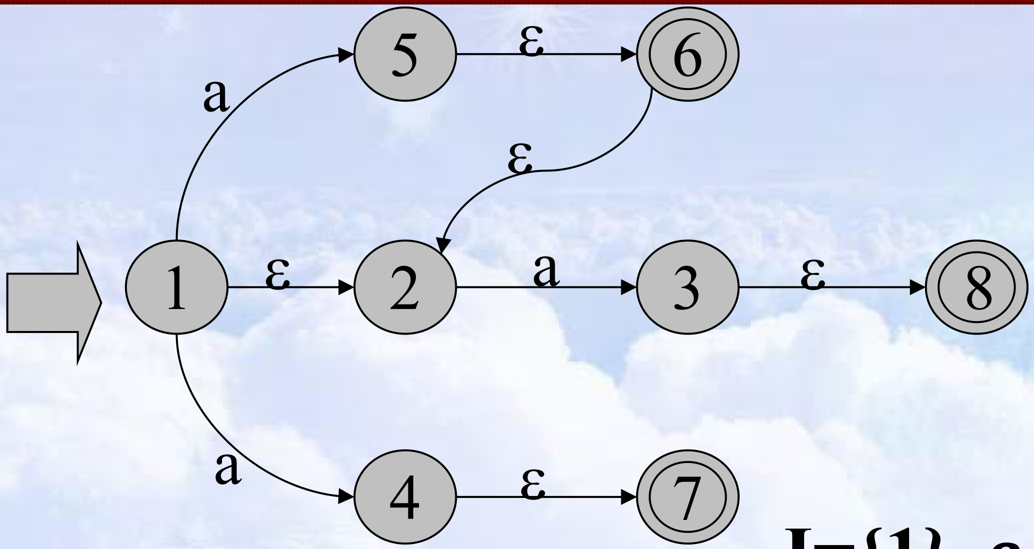
定义对状态集合I的几个有关运算

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。

状态集合I的有关运算的例子



I={1}, ε-closure(I)=

I={5}, ε-closure(I)=

move({1,2},a)=

ε-closure({5,3,4})=

NFA确定化算法:

假设NFA N=(K, $\sum$, f, K_0, K_t)按如下办法构造一个DFA M=(S, $\sum$, 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；

2. M和N的输入字母表是相同的，即是∑;

3. 转换函数是这样定义的：

$$d([S_1\ S_{2,}\ldots S_j],a)=[R_1R_2\ldots R_t] \quad \text{其中}$$

$$\{R_{1,}R_{2,}\ldots{}_{,}R_t\} = \varepsilon\text{-closure}(move(\{S_1, S_{2,},\ldots S_j\},a))$$

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

5. $S_t=\{[S_i\ S_k\ldots S_e]$，其中$[S_i\ S_k\ldots S_e]\in S$且 $\{S_i , S_{k,},\ldots S_e\}\cap K_t \neq \Phi\}$

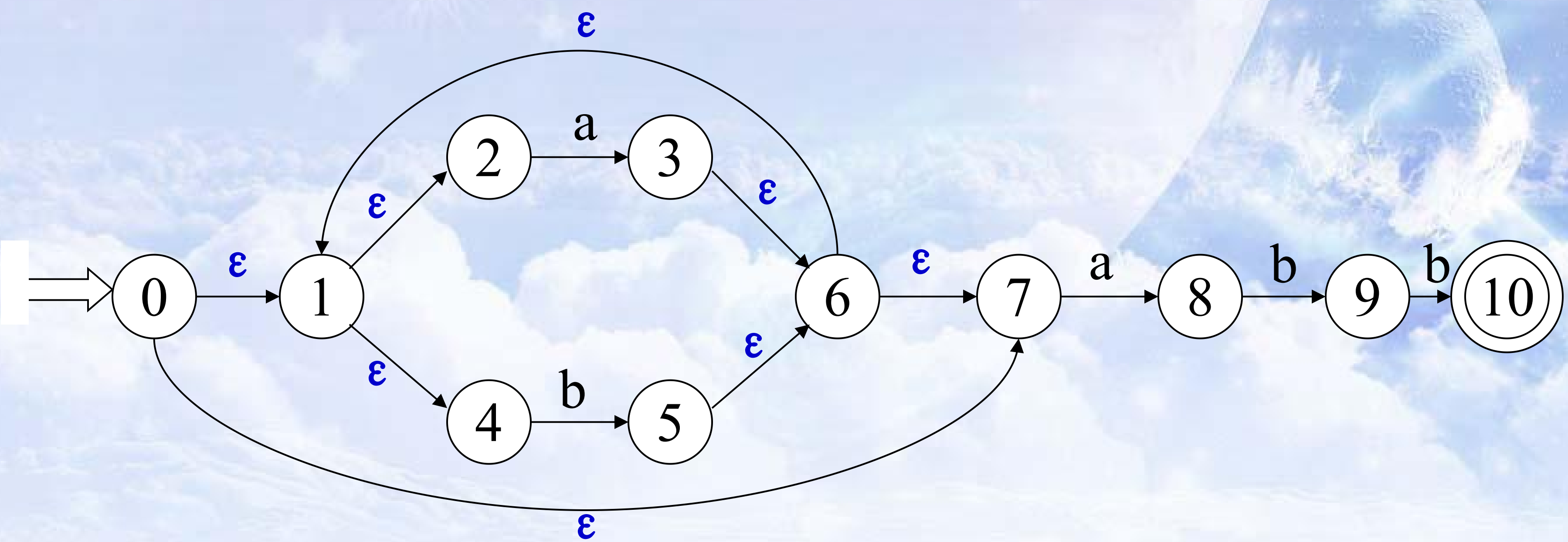
构造NFA N的状态K的子集的算法：

假定所构造的子集族为C，即C=（T_1，$T_{2,}$，... T_I），其中T_1，$T_{2,}$，... T_I为状态K的子集。

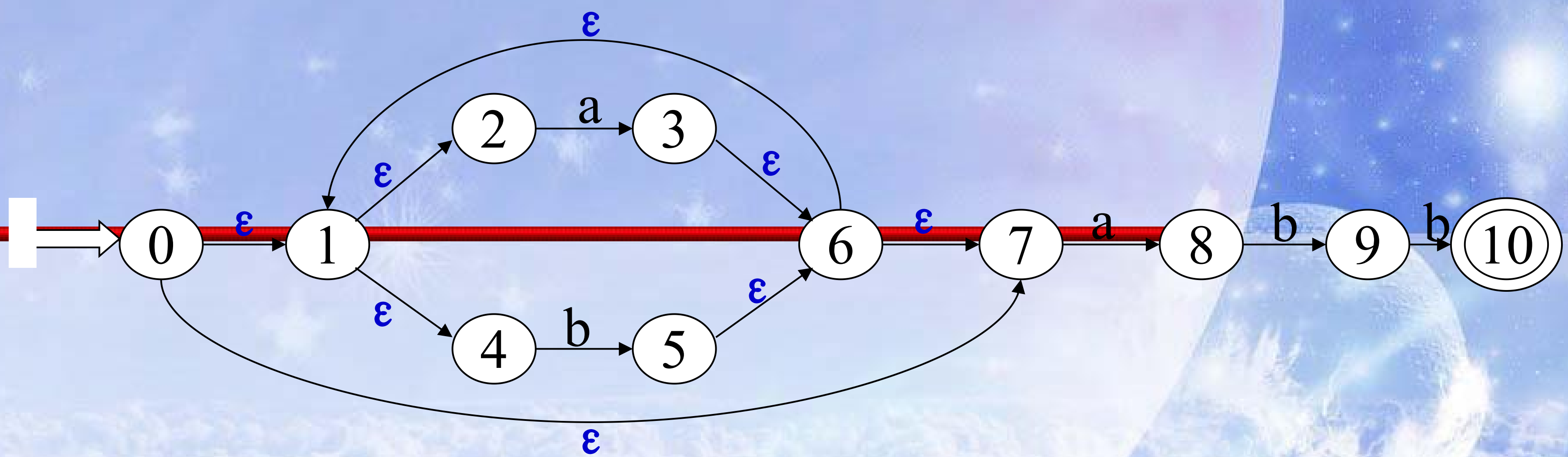
1. 开始，令ε-closure(K_0)为C中唯一成员，并且它是未被标记的。

```
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确定化举例



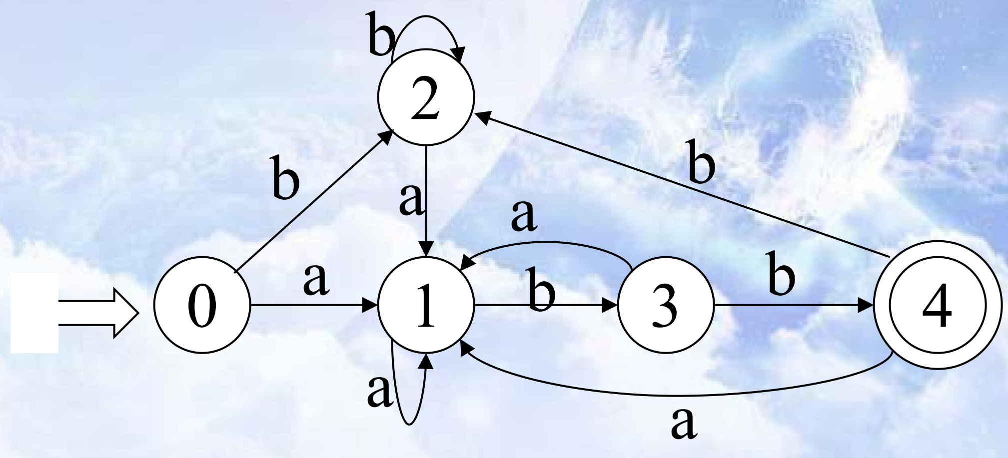
NFA N



	ε-closure(move(T_i,a))	ε-closure(move(T_i,b))	
T_0= ε-closure(0) ={0,1,2,4,7} =>0	{1,2,3,4,6,7,8}加入为T_1 1	{1,2,4,5,6,7} 加入为T_2 2	0
T_1= {1,2,3,4,6,7,8} 1	{1,2,3,4,6,7,8} 已存在T_1 1	{1,2,4,5,6,7,9} 加入为T_3 3	0
T_2= {1,2,4,5,6,7} 2	{1,2,3,4,6,7,8} 已存在T_1 1	{1,2,4,5,6,7} 已存在T_2 2	0
T_3= {1,2,4,5,6,7,9} 3	{1,2,3,4,6,7,8} 已存在T_1 1	{1,2,4,5,7,10} 加入为T_4 4	0
T_4= {1,2,4,5,7,10} 4	{1,2,3,4,6,7,8} 已存在T_1 1	{1,2,4,5,6,7} 已存在T_2 2	1

最终生成DFA

状态	a	b	
0	1	2	0
1	1	3	0
2	1	2	0
3	1	4	0
4	1	2	1



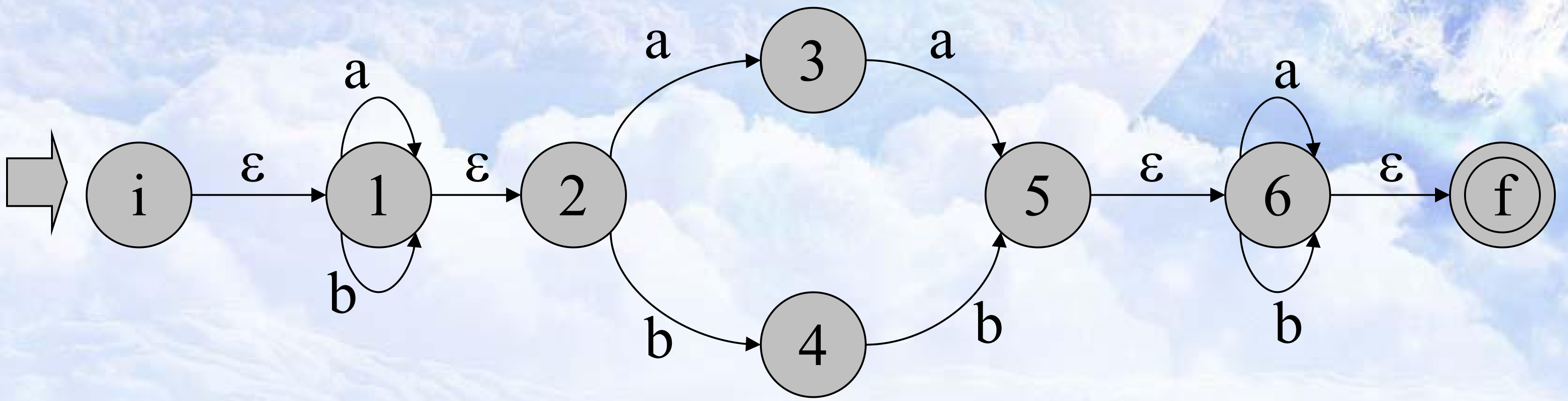
DFA M

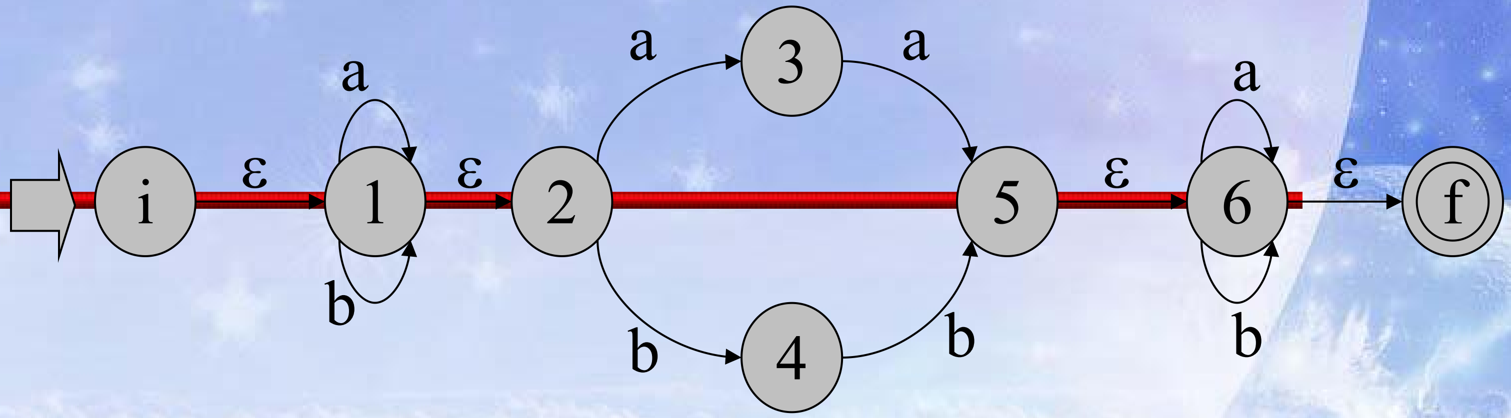
复 习

- 单词的描述工具
  - 正规文法
  - 正规式
  - 有穷自动机
- 正规文法和正规式的等价性
- 不确定的有穷自动机的确定化

NFA的确定化

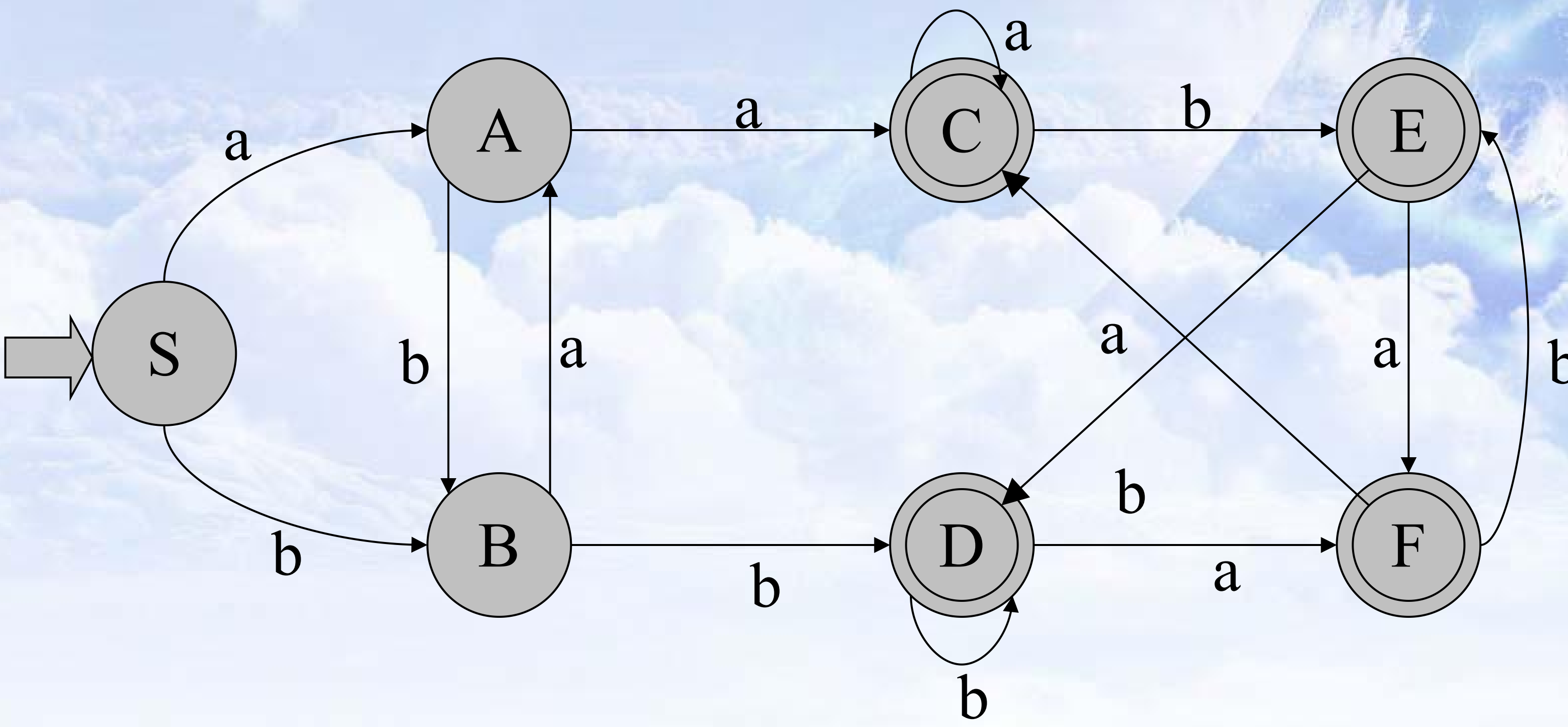
例子





	Ia	Ib	
{i,1,2} S	{1,2,3} A	{1,2,4} B	0
{1,2,3} A	{1,2,3,5,6,f} C	{1,2,4} B	0
{1,2,4} B	{1,2,3} A	{1,2,4,5,6,f} D	0
{1,2,3,5,6,f} C	{1,2,3,5,6,f} C	{1,2,4,6,f} E	1
{1,2,4,5,6,f} D	{1,2,3,6,f} F	{1,2,4,5,6,f} D	1
{1,2,4,6,f} E	{1,2,3,6,f} F	{1,2,4,5,6,f} D	1
{1,2,3,6,f} F	{1,2,3,5,6,f} C	{1,2,4,6,f} E	1

等价的DFA



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路能到达终态。

消除多余状态举例

	0	1	
s0	s1	s5	0
s1	s2	s7	1
s2	s2	s5	1
s3	s5	s7	0
s4	s5	s6	0
s5	s3	s1	0
s6	s8	s0	1
s7	s0	s1	1
s8	s3	s6	0

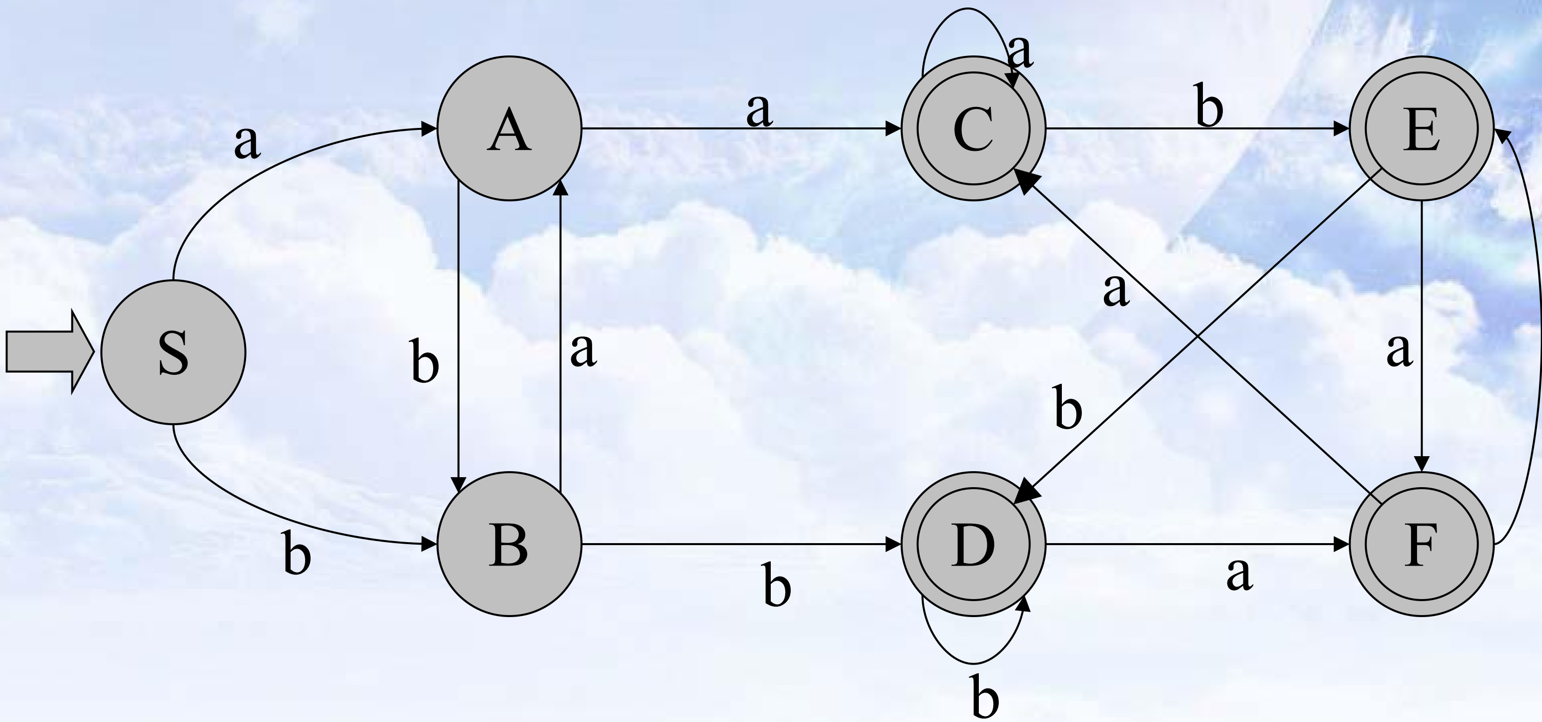
	0	1	
s0	s1	s5	0
s1	s2	s7	1
s2	s2	s5	1
s3	s5	s7	0
s5	s3	s1	0
s6	s8	s0	1
s7	s0	s1	1
s8	s3	s6	0

	0	1	
s0	s1	s5	0
s1	s2	s7	1
s2	s2	s5	1
s3	s5	s7	0
s5	s3	s1	0
s7	s0	s1	1

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等价的条件是:
  - 一致性条件——同时为终态或者非终态
  - 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。

等价状态举例



C、F等价，D、E等价，C、D等价

最小状态DFA

对于一个DFA M =（K, Σ, 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

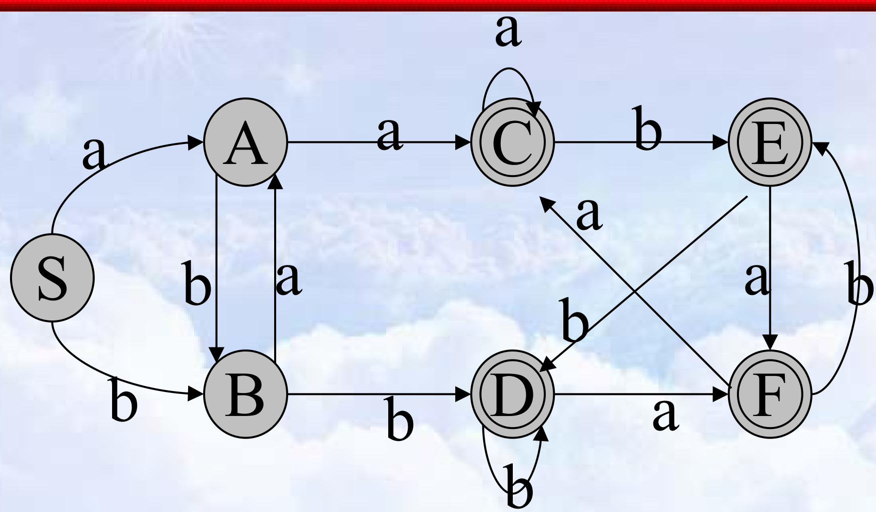
- 接受L的最小状态有穷自动机不计同构是唯一的。

"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**.

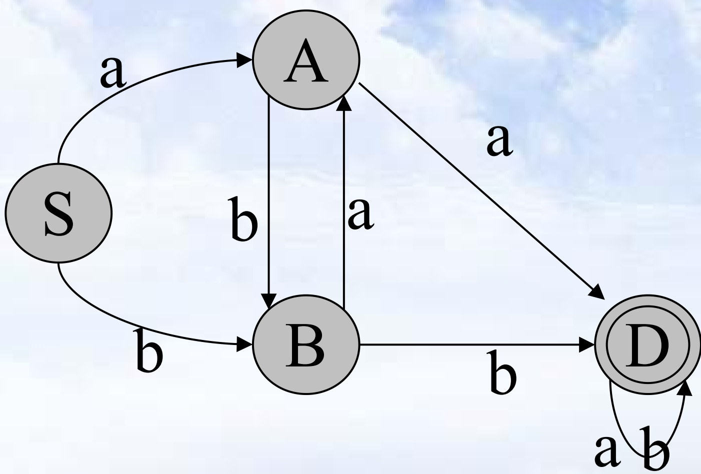
DFA的最小化—例子



终态/非终态：{S, A, B}　{C, D, E, F}

输入a：{A}　{S, B}　{C, D, E, F}

输入b：　{A}　{S}　{B}　{C, D, E, F}



DFA的最小化算法

DFA M =（K,∑,f, k_0, k_t),最小状态DFA M'

1.构造状态的一初始划分∏：

终态k_t 和非终态K- k_t两组(group)；

2.对∏施用过程PP构造新划分$\prod_{new}$

PP过程是这样的：

①将∏中的每一组G划分为子组，而G中的两个状态k_i和k_j在同一子组中的充分必要条件是：对所有输入符号a而言，k_i和k_j转换后的状态都处于∏中同样的组里;

②形成的子组构成了新划分$\prod_{new}$中的G。

3. 如Π_{new} =Π, 则令 Π_{final}=Π 并继续步骤4, 否则Π:=Π_{new}重复2。

4. 为Π_{final}中的每一组选一代表，这些代表构成M’的状态。若k是一代表且f(k, a)=t, 令r是t组的代表，则M’中有一转换f’(k, 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F的那组的代表；

5. 去掉M’中的死状态。

3.5 正规式和有穷自动机的等价性

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)。

有穷自动机→正规式

问题描述：

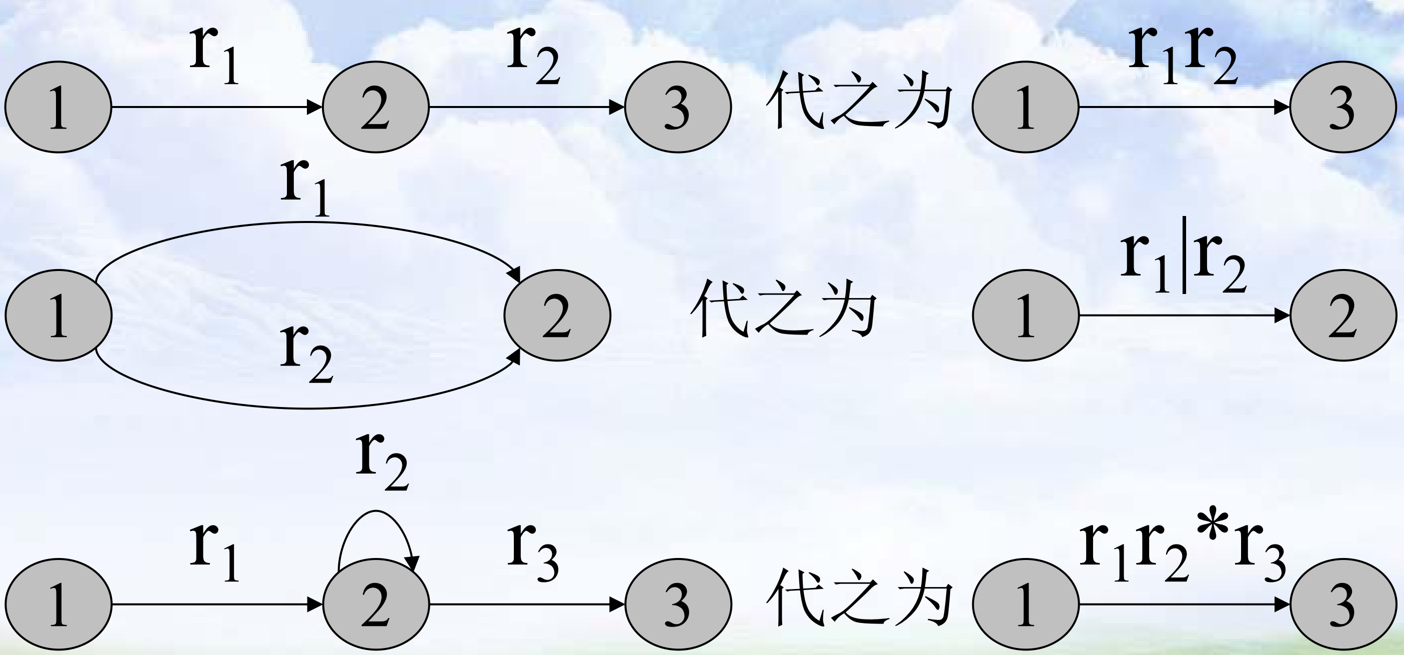
如何为Σ上的NFA M构造相应的正规式r

算法：

把状态转换图的概念拓广，令每条弧可用一个正规式作标记。

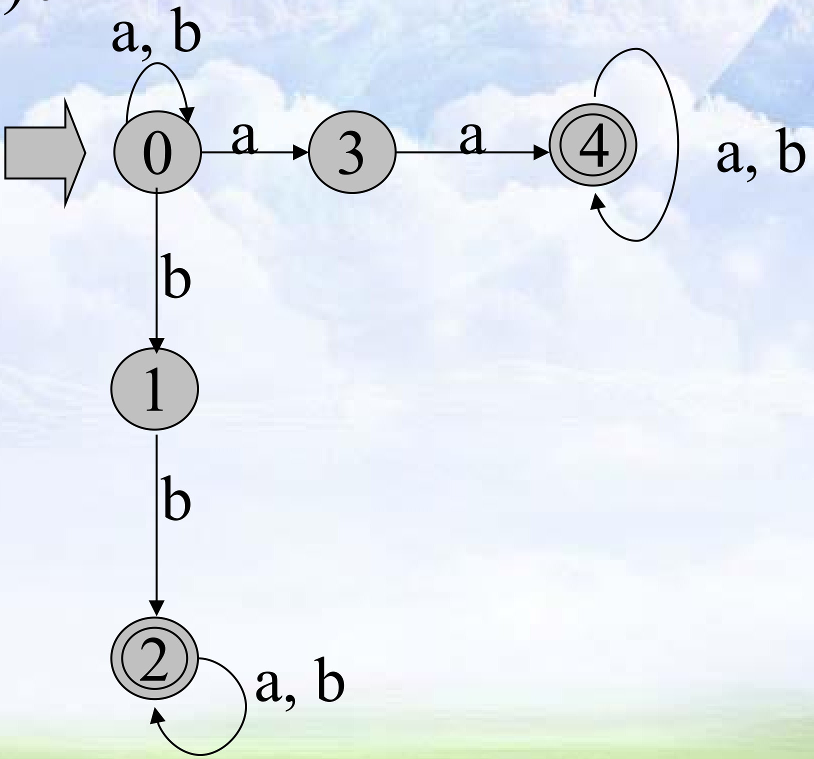
1. 在M的状态图上加进两个结点，一个为x结点，一个为y结点。从x结点用ε弧连接到M的所有初态结点，从M的所有终态结点用ε弧连接到y结点。形成一个与M等价的M’，M’只有一个初态x和一个终态y。

2. 逐步消去M’中的所有结点，直至只剩下x和y结点。在消结过程中，逐步用正规式来标记弧。其消结的规则如下：

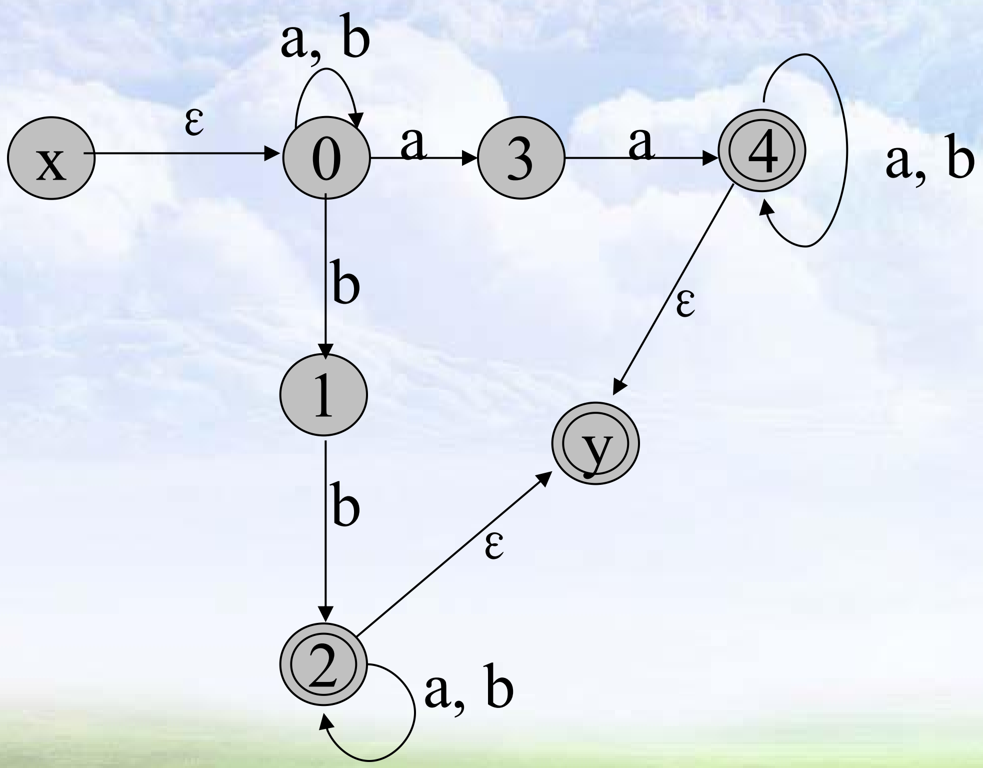


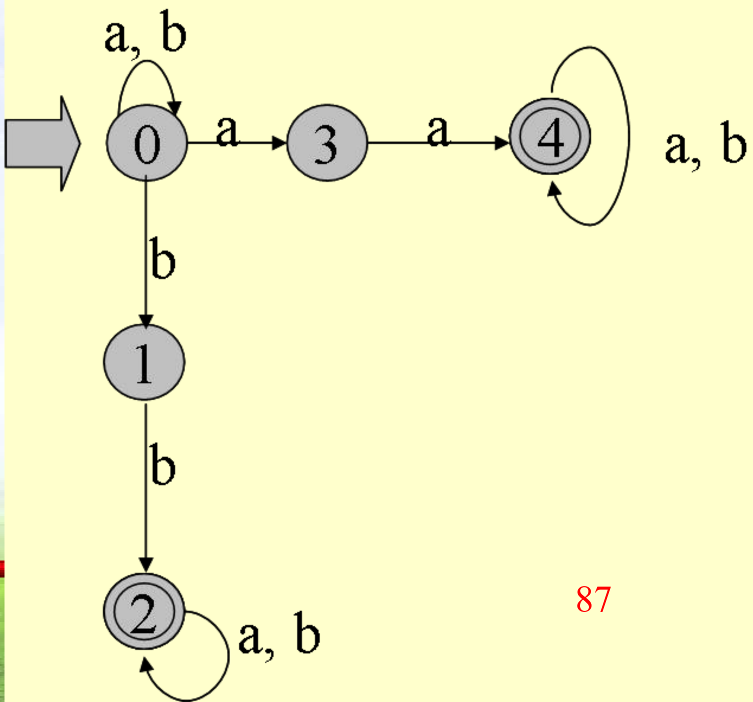
3. 最后x和y结点间的弧上的标记则为所求的正规式r。

例：将下图中的NFA M，求正规式r，使得L(r)=L(M)。

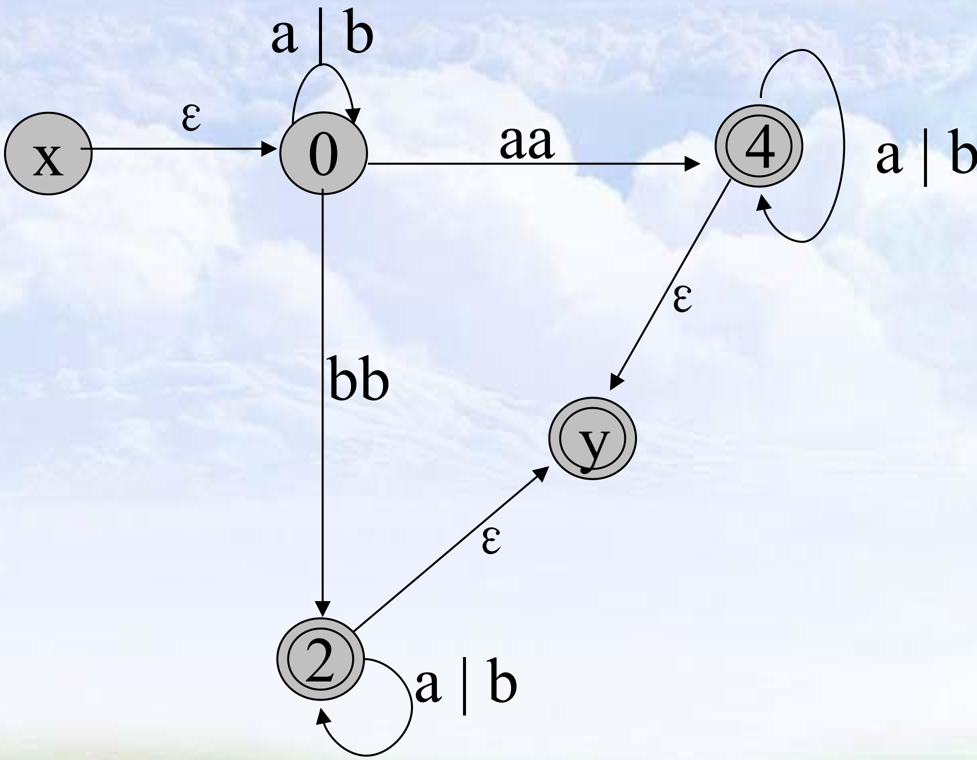


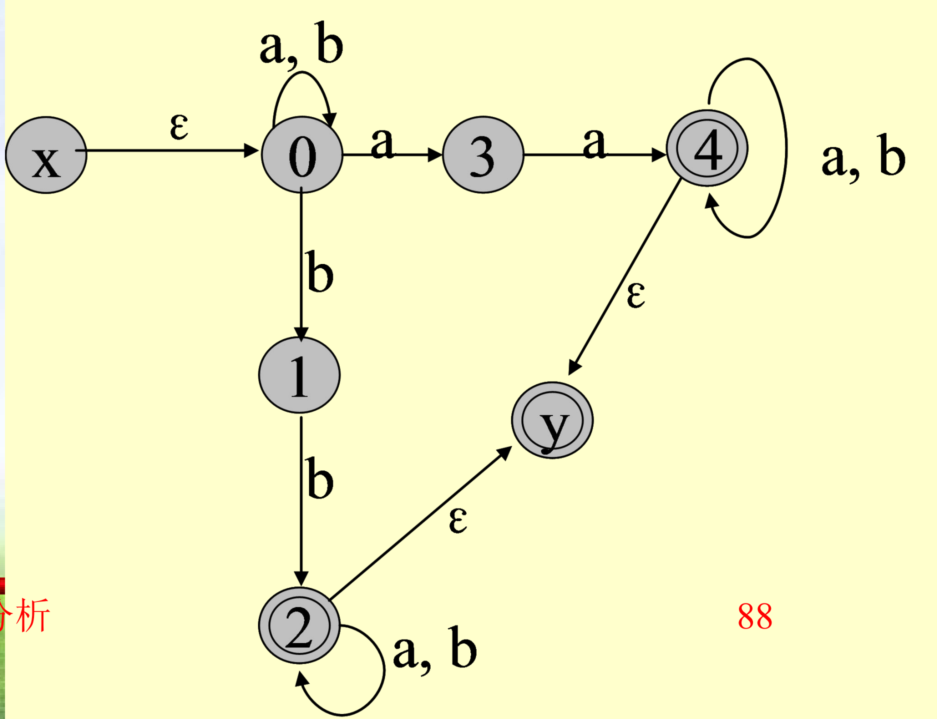
1. 加x和y结点，形成如下的自动机M’



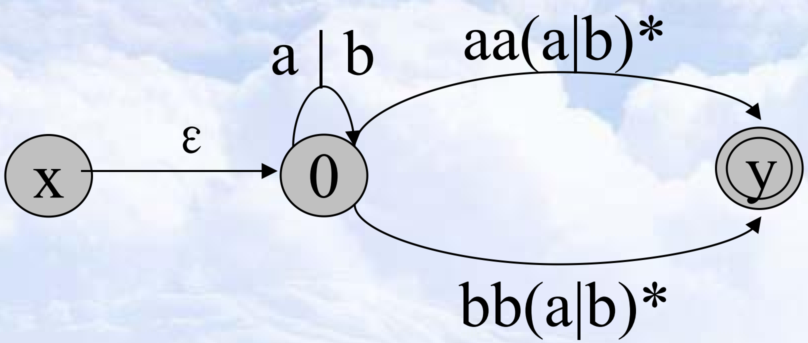


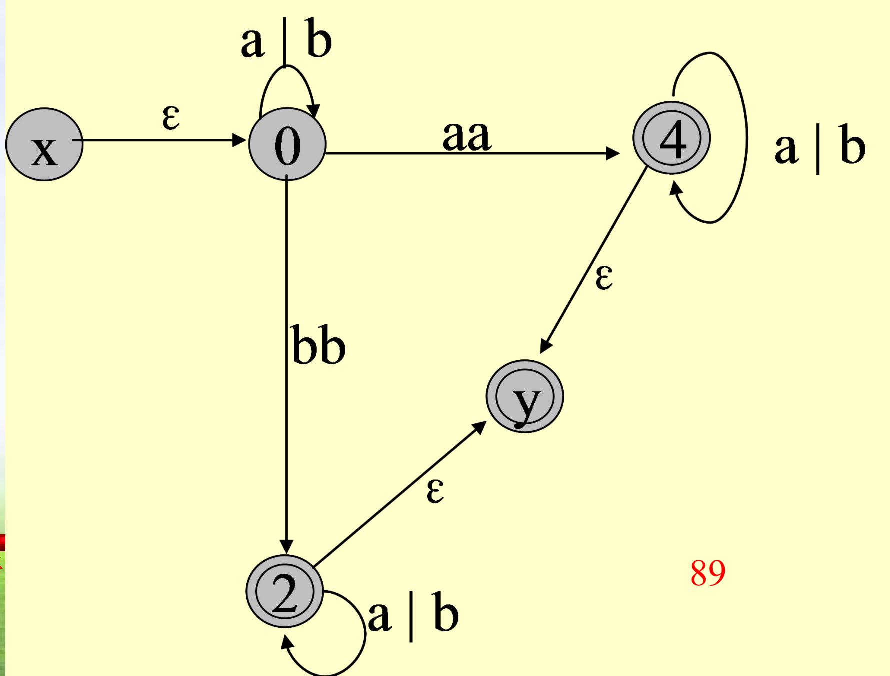
2. 消去M’中的1和3结点



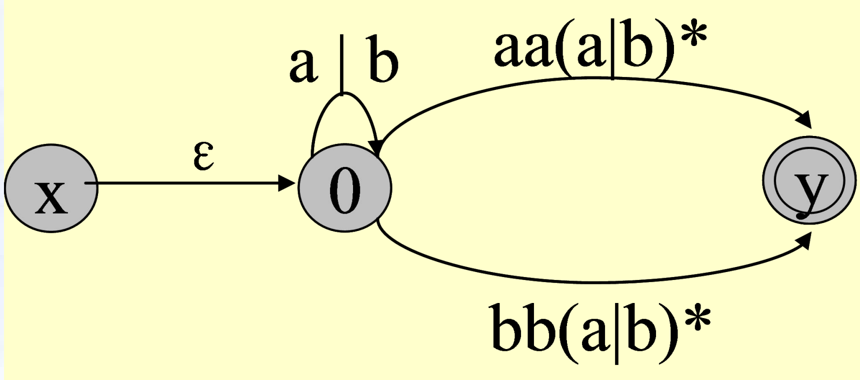


3. 消去2和4结点





4. 消去0结点，只剩下x和y结点如下图：



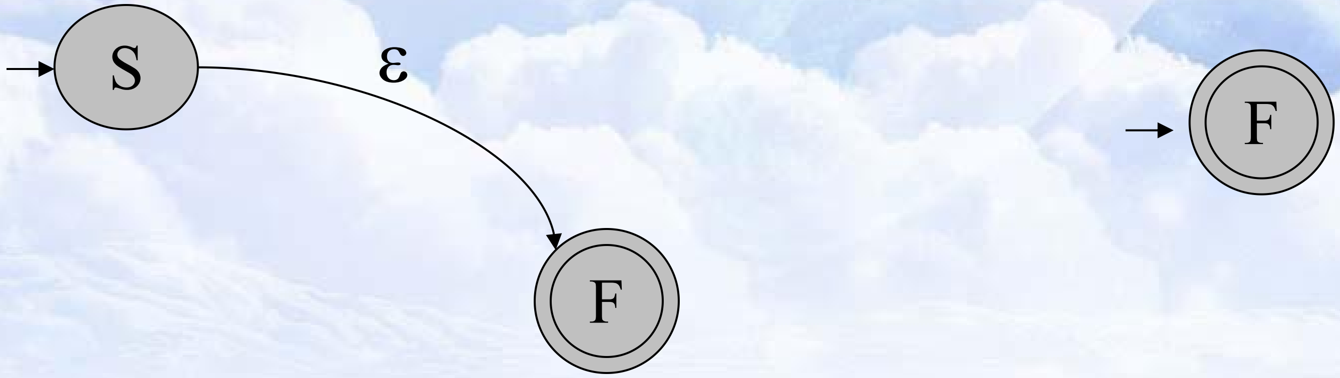
r=(a|b)*(aa|bb)(a|b)*即为所求。

正规式→有穷自动机

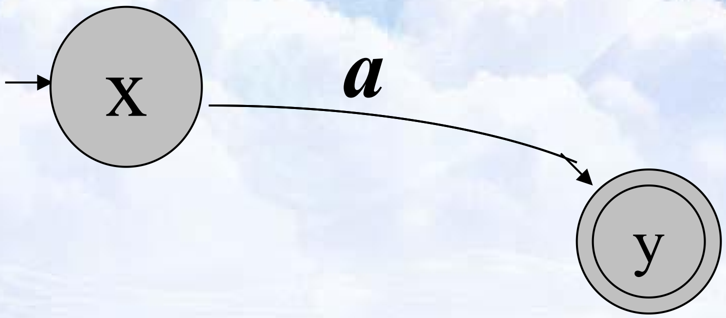
- 从Σ上的一个正规式R构造Σ上的一个NFA M，使得L(M)=L(R)的方法。
- “语法制导”的方法，即按正规式的语法结构指引构造过程，构造规则具体描述如下：

对于正规式R=∅ ,构造的NFA

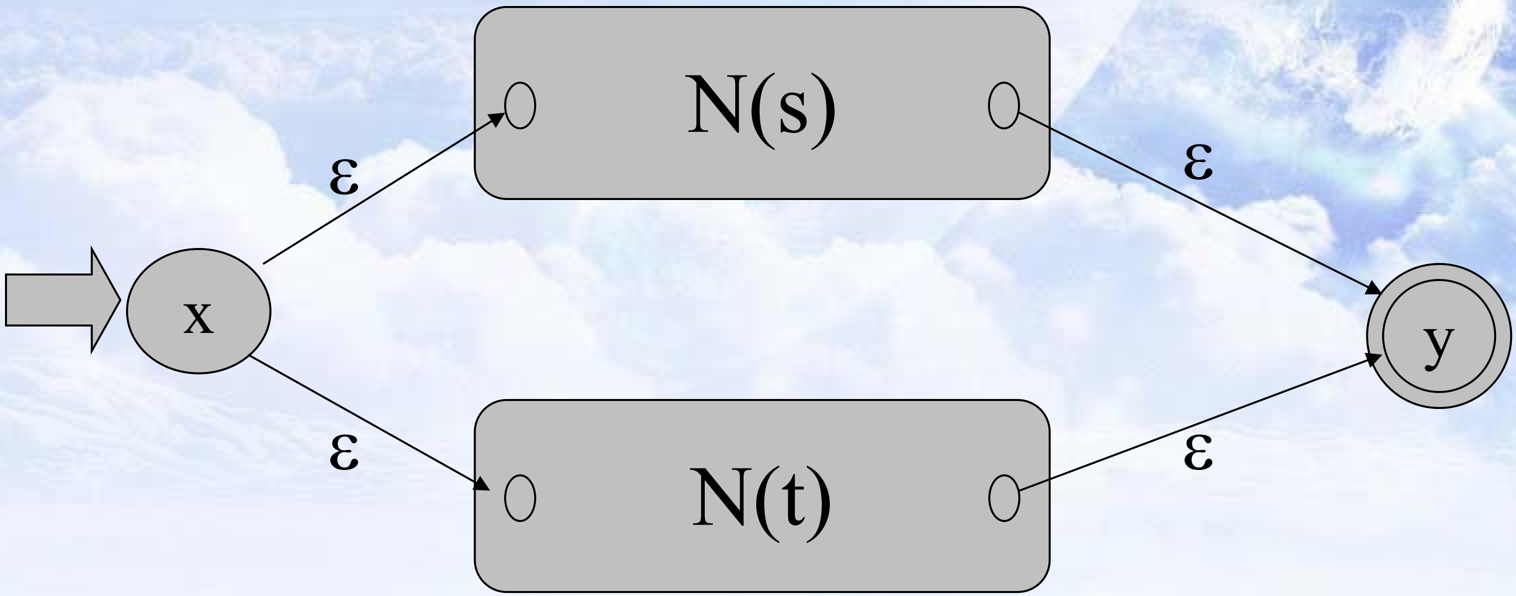
对于正规式ε ,构造的NFA（二种）



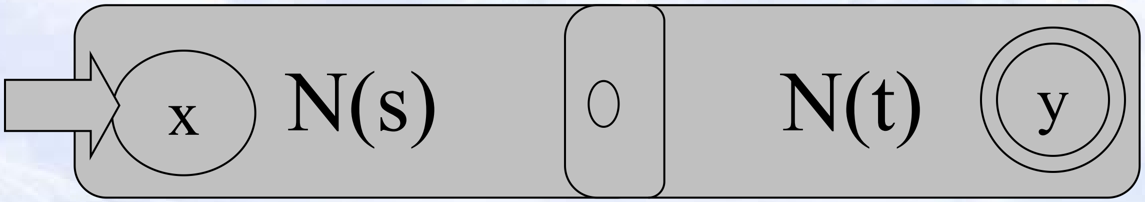
对于正规式a, a $\in \Sigma$ 构造的NFA



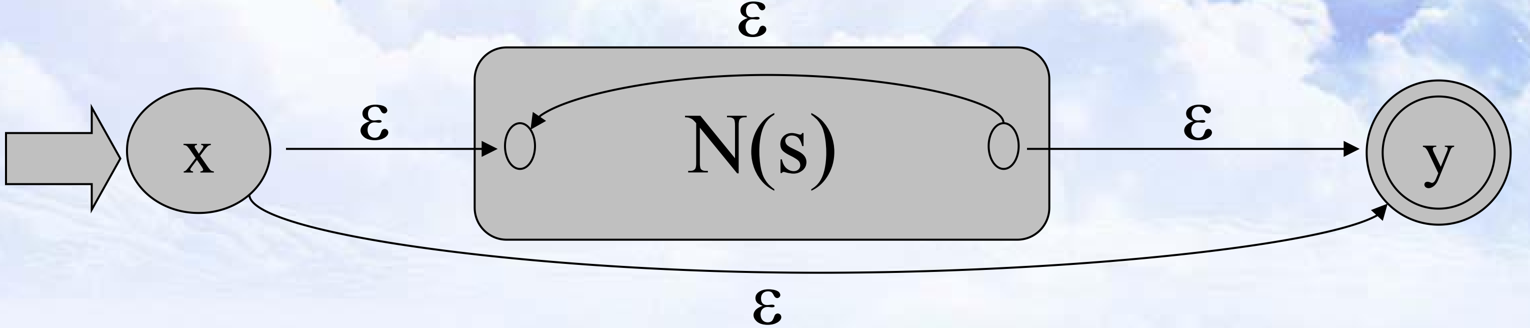
对于正规式r, r= s|t构造的NFA



对于正规式r, r=st构造的NFA



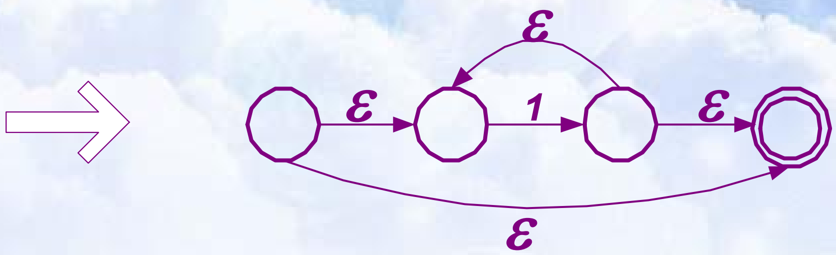
对于正规式r, r=s*构造的NFA



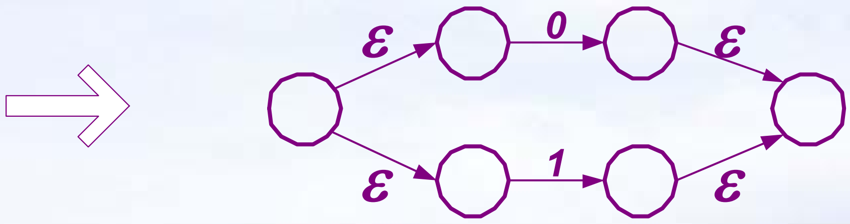
举例:从正规表达式构造等价的ε - NFA

正规表达式 1*0(0 | 1)*

1*

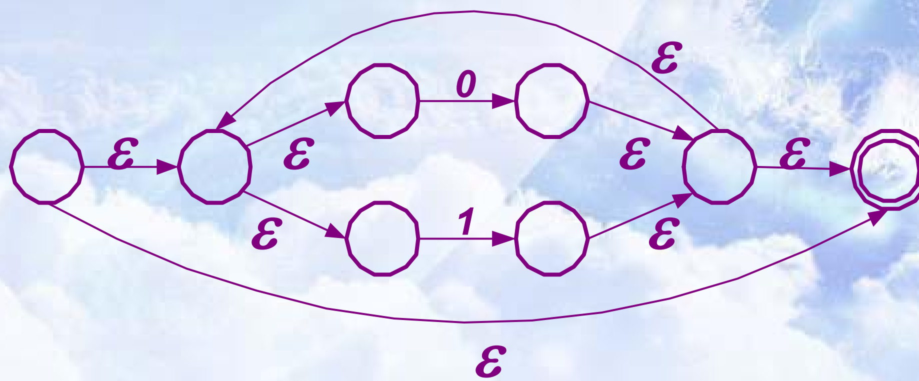


0 | 1

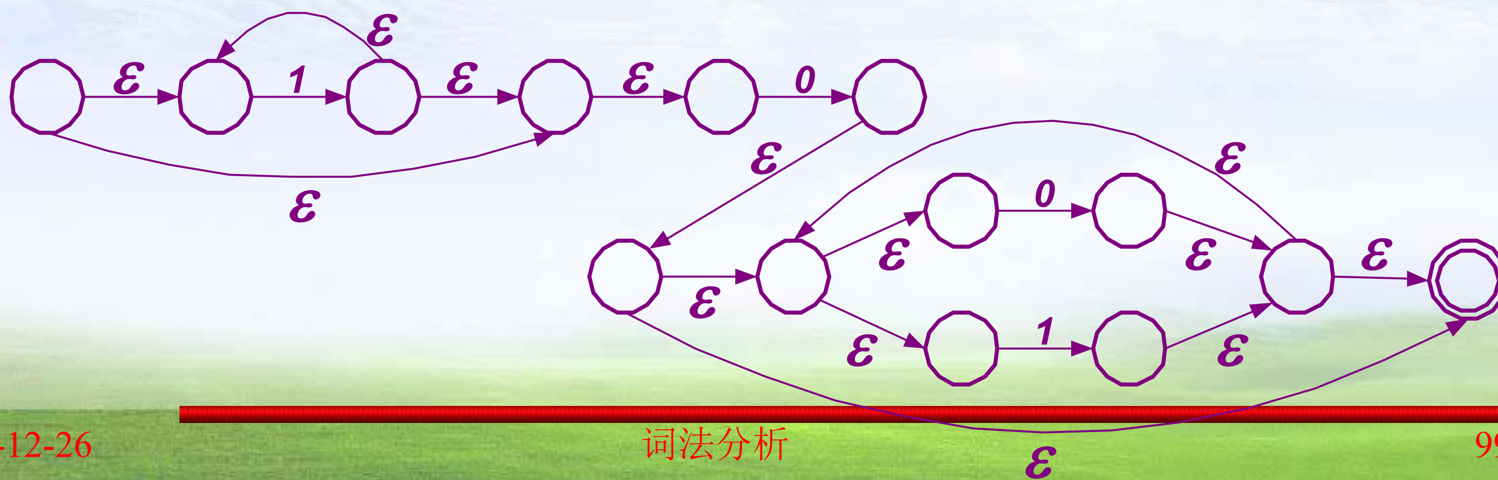


从正规表达式构造等价的ε - NFA

*(0 | 1)** ⇒

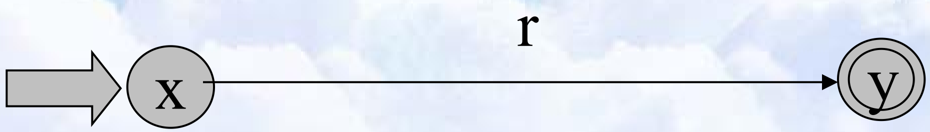


*1*0(0 | 1)** ⇒

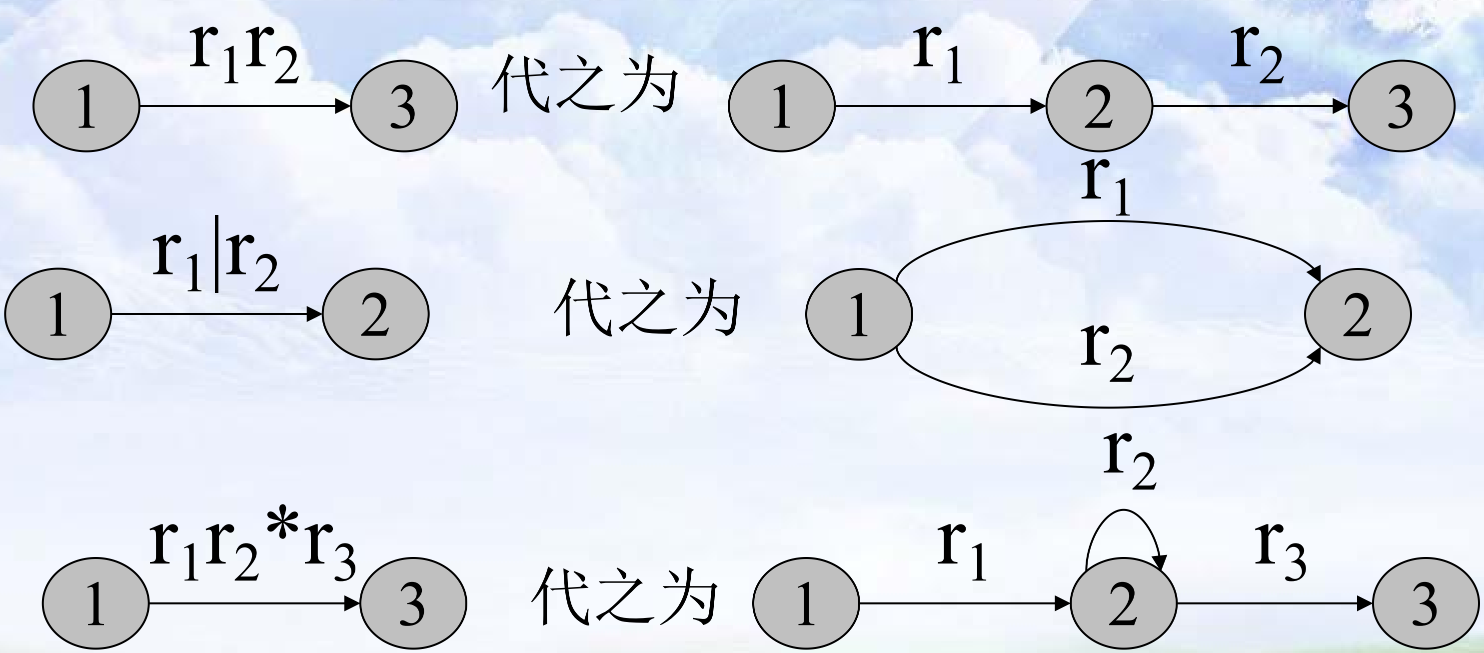


补充：正规式→有穷自动机的简易方法

1. 对于正规式r，生成有穷自动机M如下：

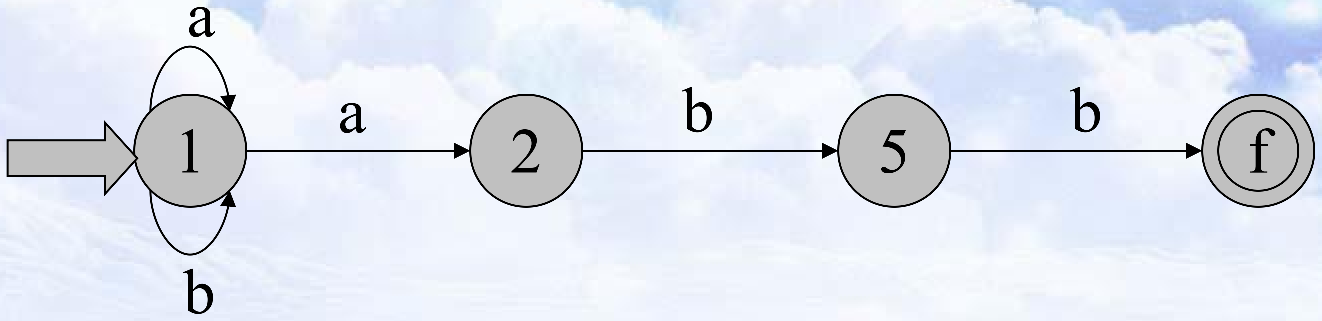


2. 逐步消去M中的所有标记的长度大于1的弧，规则如下：



3. 所有弧上的标记都为a($a \in V_N$)或者ε。所得到的NFA即为所求。

例：R=(a|b)* abb



3.6 正规文法和有穷自动机的等价性

采用下面的规则可以从正规文法G直接构造一个有穷自动机NFA M，使得L(M)=L(G)。

1. **M的字母表与G的终结集相同；**
2. **为G中的每个非终结符生成M的一个状态，G的开始符号S是开始状态S；**
3. **对G中的形如A→tB的规则(其中t为终结符号或ε，A和B为非终结符的产生式)，构造M的一个转换函数f(A, t)=B；**
4. **增加一个新状态Z，作为NFA的终态；**
5. **对G中的形如A→t的产生式，构造M的一个转换函数f(A, t)=Z。**

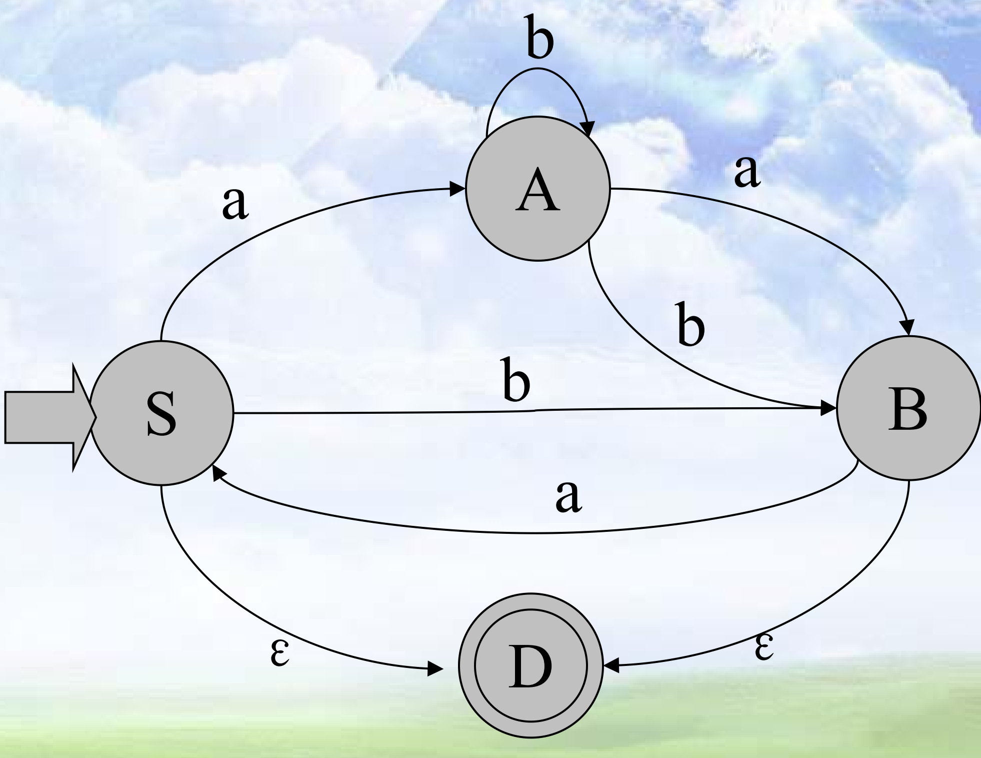
例：

构造与下述文法G[S]等价的NFA M。

G[S]: S→aA |bB | ε

A →aB | bA

B → aS | bA | ε



有穷自动机→正规文法

对转换函数f(A, t)=B可写成一个产生式：

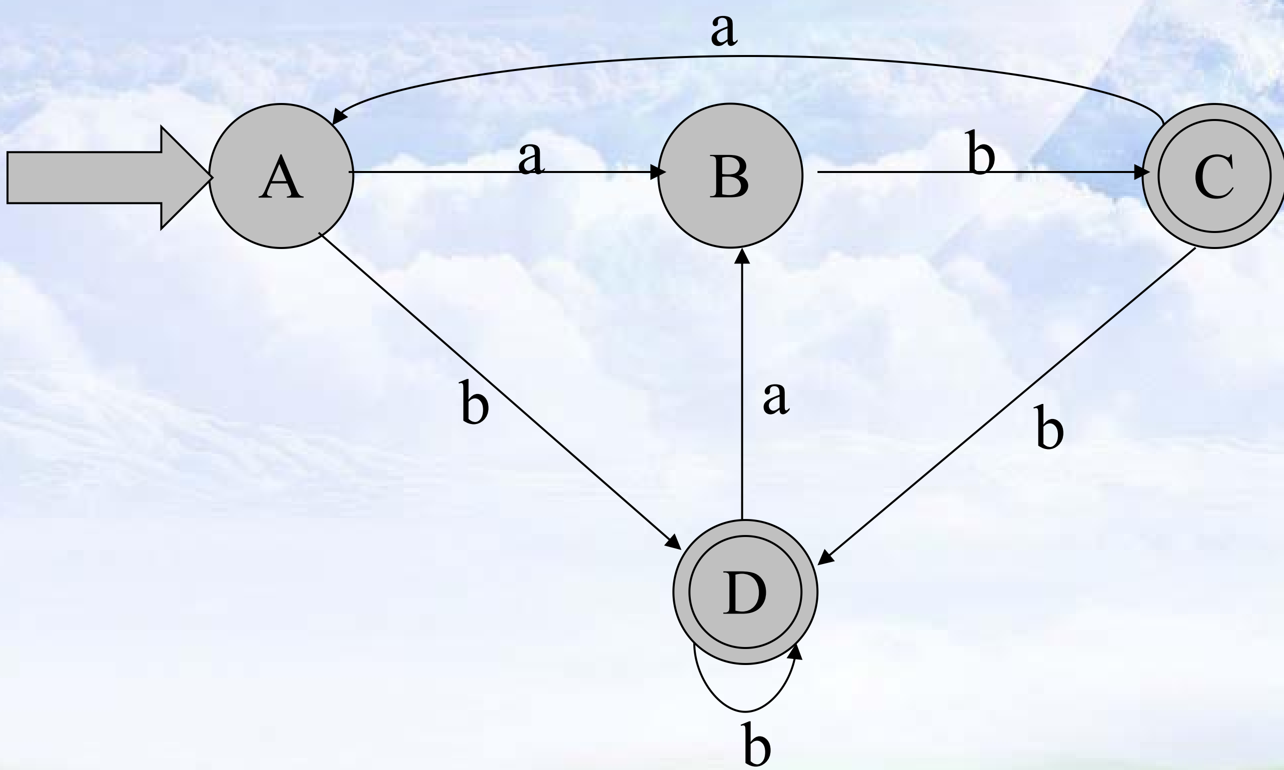
A→tB

对可接受状态Z，增加一产生式：

Z→ ε

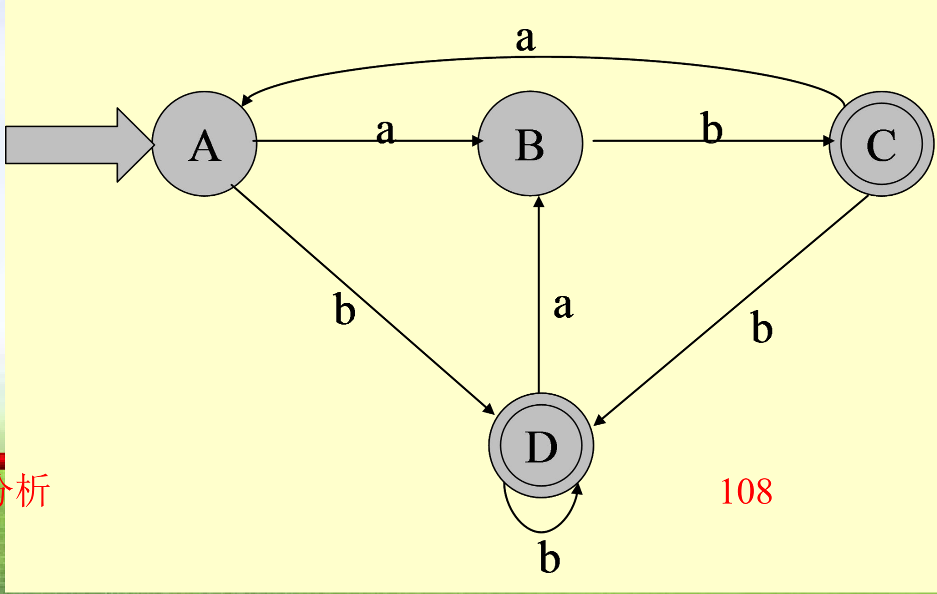
例：

给出与下图的NFA等价的正规文法G



等价文法G=({A, B, C, D}, {a, b}, P, A)，其中P为：

A→aB
A→bD
B→bC
C→aA
C→bD
C→ ε
D→aB
D→bD
D→ ε



词法分析程序的应用

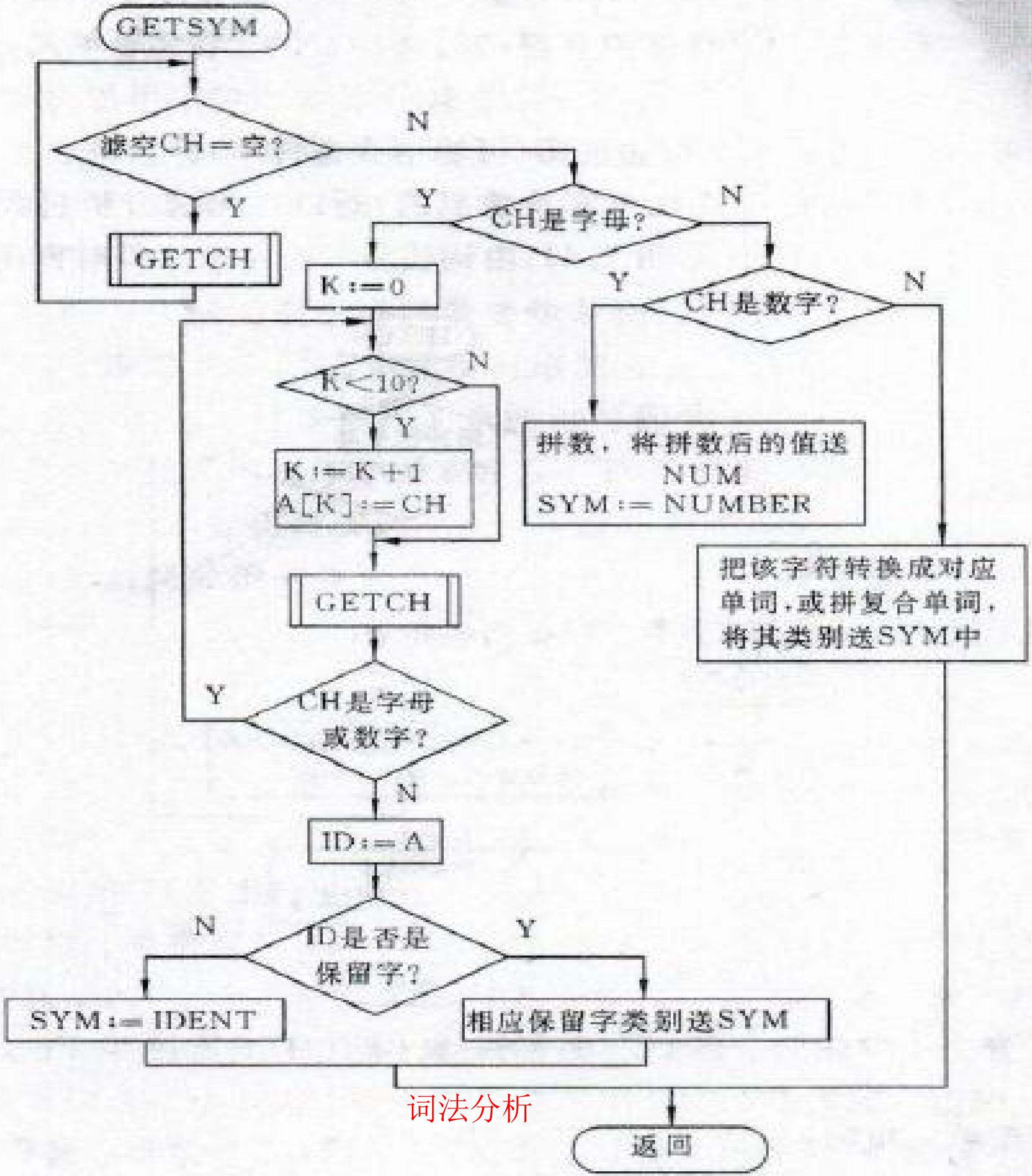
- 词法分析程序的设计技术可应用于其它领域，比如查询语言以及信息检索系统等，这种应用领域的程序设计特点是，通过字符串模式的匹配来引发动作。
- 词法分析程序的自动构造工具也广泛应用于许多方面，如用以生成一个程序，可识别印刷电路板中的缺陷，又如开关线路设计和文本编辑的自动生成等。

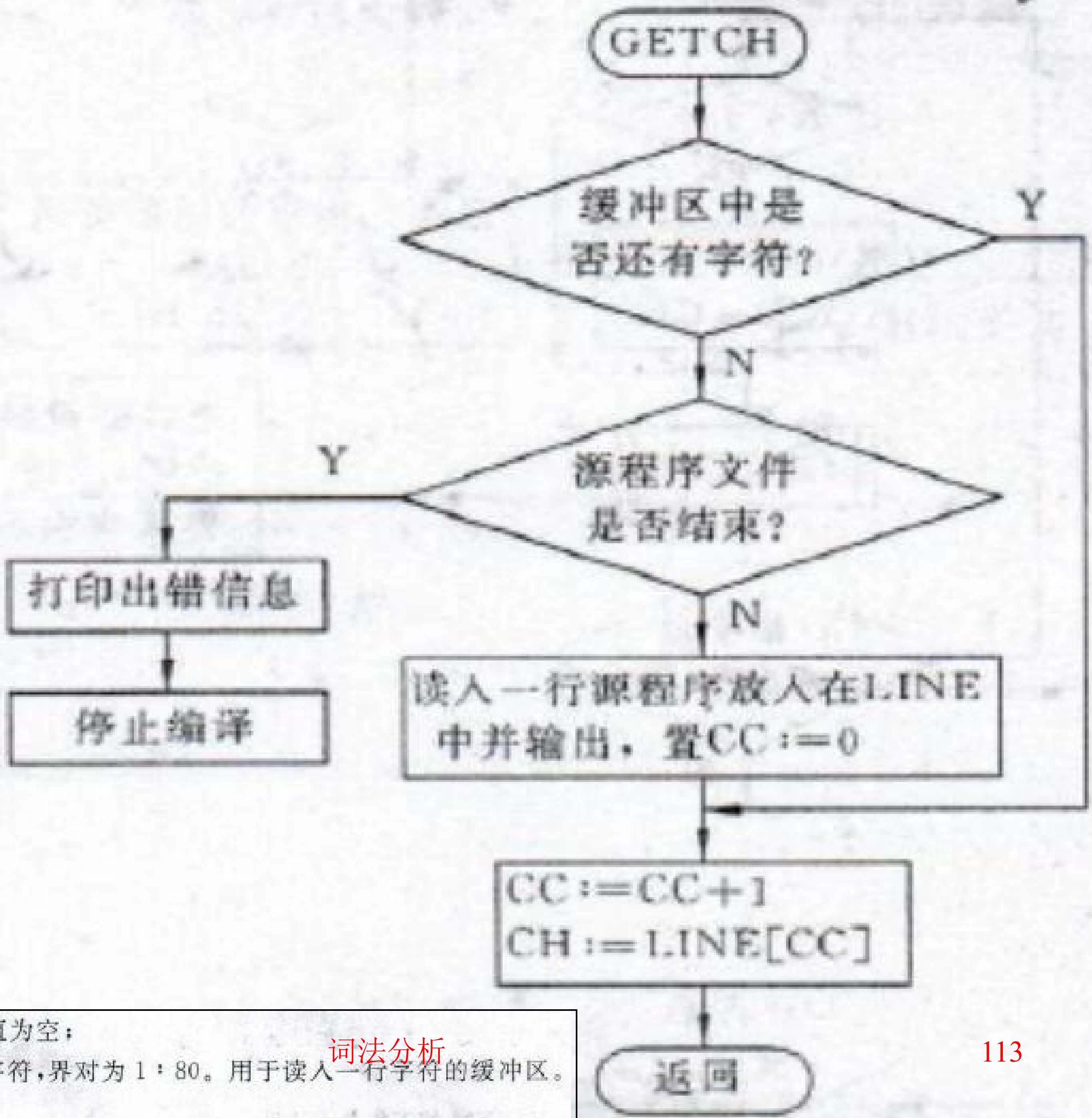
3.7 PL/0词法分析程序

- **识别的单词：**
  - **保留字：如：BEGIN、 END、 IF、 THEN等**
  - **运算符： 如：+、-、*、/、：=、#、>=、<=等**
  - **标识符： 用户定义的变量名、常数名、过程名**
  - **常数： 如：10、25、100等整数**
  - **界符： 如： ‘,’、 ‘.’、 ‘;’、 ‘(’、 ‘)’等**

词法分析过程GETSYM所要完成的任务：

- 从源程序读字符（getch）
- 滤空格
- 识别保留字
- 识别标识符
- 拼数
- 识别单字符单词
- 拼双字符单词


GETSYM
滤空CH＝空?
N
Y
GETCH
Y
CH是字母?
N
K:=0
Y
CH是数字?
N
K<10?
N
Y
K:=K+1
A[K]:=CH
拼数，将拼数后的值送
NUM
SYM:=NUMBER
把该字符转换成对应
单词，或拼复合单词，
将其类别送SYM中
GETCH
Y
CH是字母
或数字?
N
ID:=A
N
ID是否是
保留字?
Y
SYM:=IDENT
相应保留字类别送SYM
返回



CH　存放当前读取的字符，初值为空；
LINE　一维数组，数组元素是字符，界对为 1 : 80。用于读入一行字符的缓冲区。
LL 和 CC 为计数器，初值为 0。

本章小结

正规文法

正规式

有穷自动机：DFA、NFA、确定化、化简

正规文法←→正规式

正规式←→有穷自动机

正规文法←→有穷自动机

习题

3

4

5

8

9