第四章 词法分析

词法分析程序亦称为扫描器

扫描器的任务是识别基本的语法单位
——单词

扫描器的输出是语法分析程序的输入

第四章 词法分析

词法分析程序的设计和实现

- 首先需要描述和刻画语言中的原子单位——单词，其次需要识别单词和执行某些相关的动作。描述程序设计语言的词法的机制是3型文法和正则表达式，识别机制是有穷状态自动机。

第四章 词法分析

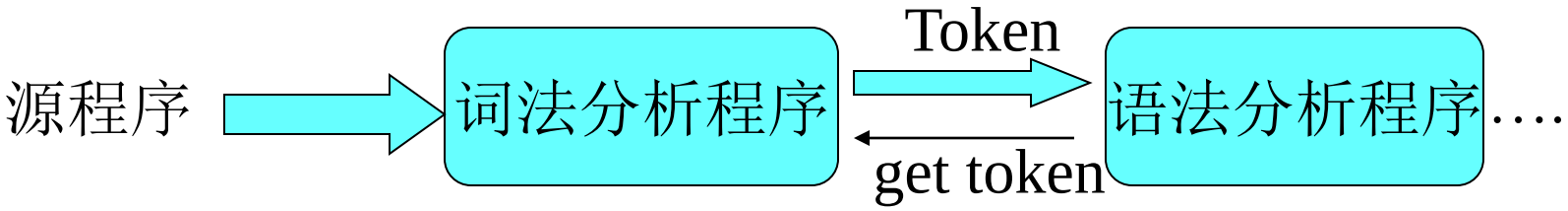
- 设计词法分析程序
- 单词的描述工具
- 单词的识别系统

- 4.1 词法分析程序
- 4.2 正规表达式与正规集（正规语言）
- 4.3 有穷自动机
- 4.4有穷自动机和正规表达式
- 4.5有穷自动机和正规文法

4.1词法分析程序

- 词法分析（lexical analysis）
  - 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。

4.1词法分析程序



- 主要任务：
  - 读源程序，产生单词符号，并转换为token表示
- 其他任务：
  - 滤掉空格，删除注释、换行符
  - 对行列计数
  - 发现并定位词法错误，并尽量改正
  - 建立符号表、常数表等表格，……

4.1词法分析程序

- 4.1.1词法分析器的输入缓冲区
- 4.1.2词法分析器的输出
- 单词符号一般可分为下列五种：
  - 基本字，关键字
  - 标识符
  - 常数（量）
  - 运算符
  - 界符

4.1词法分析程序

- 输出的二元式（token）表示：
- （单词种别，单词自身的值）
- Token的种类：
- 1、有些单词，只需要值，如基本字；
- 2、有些单词，还需要其他信息，如标示符。
- 例： A:=B+2
  (Id的整数码,指向A的符号表的入口指针）
  (运算符的整数码,’:=’)
  (Id的整数码,指向B的符号表的入口指针)
  (运算符的整数码,’+’)
  (常数的整数码, 2)
- 4.1.3 以状态转换图为例设计词法分析器

4.2正规表达式与正规集（正规语言）

- 正规表达式（regular expression）是描述单词符号的一种方便工具，也是定义正规集的工具。
- 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，它们所表示的正规集分别为{ε}和{ }；

4.2正规表达式与正规集（正规语言）

- **2)**任何$a \in \Sigma$，a是Σ上的一个正规式，它所表示的正规集为{a}；
- **3)**假定e_1和e_2都是Σ上的正规式，它们所表示的正规集分别为$L(e_1)$和$L(e_2)$，那么，(e_1), $e_1 \mid e_2$, $e_1 \bullet e_2$, 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字集才是Σ上的正规集。

4.2正规表达式与正规集（正规语言）

- 其中的“|”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，
- 规定算符的优先顺序为：

“(”、“)”、“*”、“•”、“|”。

高 低

- 连接符“•”一般可省略不写。
- “*”、“•”和“|”都是左结合的。

4.2正规表达式与正规集（正规语言）

- 例1 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}

4.2正规表达式与正规集（正规语言）

- 正规式　　正规集
- (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}

4.2正规表达式与正规集（正规语言）

- 例2 Σ={l，d}，r=l(l｜d) *定义的正规集？
- {l,ll,ld,ldd,……}
- 例3 Σ={d，.，e，+，-},则Σ上的正规式d*(.dd *｜ε)(e(+｜-｜ε)dd*｜ε)表示的是无符号数的集合。其中d为0~9的数字。

4.2正规表达式与正规集（正规语言）

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

4.2正规表达式与正规集（正规语言）

- 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

4.2正规表达式与正规集（正规语言）

– 5、 $\varepsilon r=r,\ r\varepsilon=r$　　ε是“连接”的恒等元素零一律

– 6、 $r|r=r$

$r^*=\varepsilon|r^+$

$r^+=rr^*$

正规文法和正规式

- 文法的定义
  - $G=\{V_N,V_T,P,S\}$
  - V_N：非终结符的非空有穷集
  - V_T：终结符的非空有穷集
  - P：　产生式的非空有穷集

    $\alpha\rightarrow\beta$　$\alpha\in(V_N\cup V_T)^*$且至少含一个非终结符，　$\beta\in(V_N\cup V_T)^*$
  - S：　$\in V_N$, 称为开始符号
- 正规文法：　G的任何产生式为$A\rightarrow aB$或$A\rightarrow a$，其中$a\in V_T\cup\varepsilon$，A，$B\in V_N$

正规文法和正规式

- G1=({S,A,B},{a,b},S,P}
  其中P：S→A
  S→B
  A→aA
  A→a
  B→bB
  B→b
- G2：S→I|IT
  T→I|d|IT|dT

正规文法和正规式

对任意一个正规文法，存在一个定义同一个语言的正规式；反之亦然。

- ∑上的正规式=>正规文法
- 初始$V_T=\Sigma, S \in V_N$，生成正规产生式(或定义式) :$S \rightarrow r$（r为正规式）
- (R1) 对形如 $A \rightarrow r_1 r_2$的正规产生式：
  $A \rightarrow r_1 B$
  $B \rightarrow r_2$
  $B \in V_N$

  (R2)对形如$A \rightarrow r^* r_1$的正规产生式：
  $A \rightarrow rB$
  $A \rightarrow r_1$
  $B \rightarrow rB$
  $B \rightarrow r_1 \quad B \in V_N$

  (R3)对形如$A \rightarrow r_1 \mid r_2$的正规产生式:
  $A \rightarrow r_1$
  $A \rightarrow r_2$
- 不断应用R做变换，直到每个产生式右端只含一个V_N

正规文法和正规式

- 例　r=a(a|d)*转换为正规文法

  V_T={a,d}　　S→a(a|d)*

  - R1　S→aA

    A→(a|d)*

  - R2　A→(a|d)B

    A→ε

    B→(a|d)B

    B→ε

正规文法和正规式

– R3 G[s]: P

$S \rightarrow aA$

$A \rightarrow \varepsilon$

$A \rightarrow aB$

$A \rightarrow dB$

$B \rightarrow aB$

$B \rightarrow dB$

$B \rightarrow \varepsilon$

$V_T=\{a,d\}$

$V_N=\{S,A,B\}$

正规文法和正规式

- 例：将标示符集合的正规式：letter(letter|digit)*转换为正规文法

  $S \rightarrow letterA$

  $A \rightarrow (letter|digit)^*$

  $\begin{cases} A \rightarrow (letter|digit)B|\ \varepsilon \\ B \rightarrow (letter|digit)B|\ \varepsilon \end{cases}$

- 结果：$\begin{cases} S \rightarrow letterA \\ A \rightarrow letter\ B|digitB|\ \varepsilon \\ B \rightarrow letterB|digitB|\ \varepsilon \end{cases}$
- 例：整数的正规式$(digit)^+$
- 运算符的正规式$relop \rightarrow <|<=|=|<>|>|>=$

正规文法和正规式

正规文法=>正规式

正规文法		正规式
– $A \rightarrow xB$, $B \rightarrow y$	转换成：	$A=xy$
– $A \rightarrow xA \mid y$	转换成：	$A=x^*y$
– $A \rightarrow x \mid y$	转换成：	$A=x \mid y$

正规文法和正规式

– G[s]:$S \to aA$
$S \to a$
$A \to aA$
$A \to dA$
$A \to a$
$A \to d$

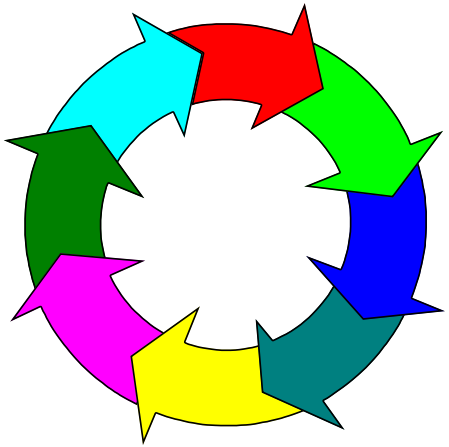
– ① $S \to aA \mid a$
② $A \to aA \mid a \mid dA \mid d$
③ $A \to (a \mid d)A \mid (a \mid d)$
④ $A \to (a \mid d)^*(a \mid d)$

正规文法和正规式

- S =a(a|d)*(a|d)|a
  =a((a|d)*(a|d)|ε)
  =a((a|d)+|ε)
- R=a(a|d)*

4.3有穷(限)自动机

- 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



4.3.1 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为输入符号字母表；

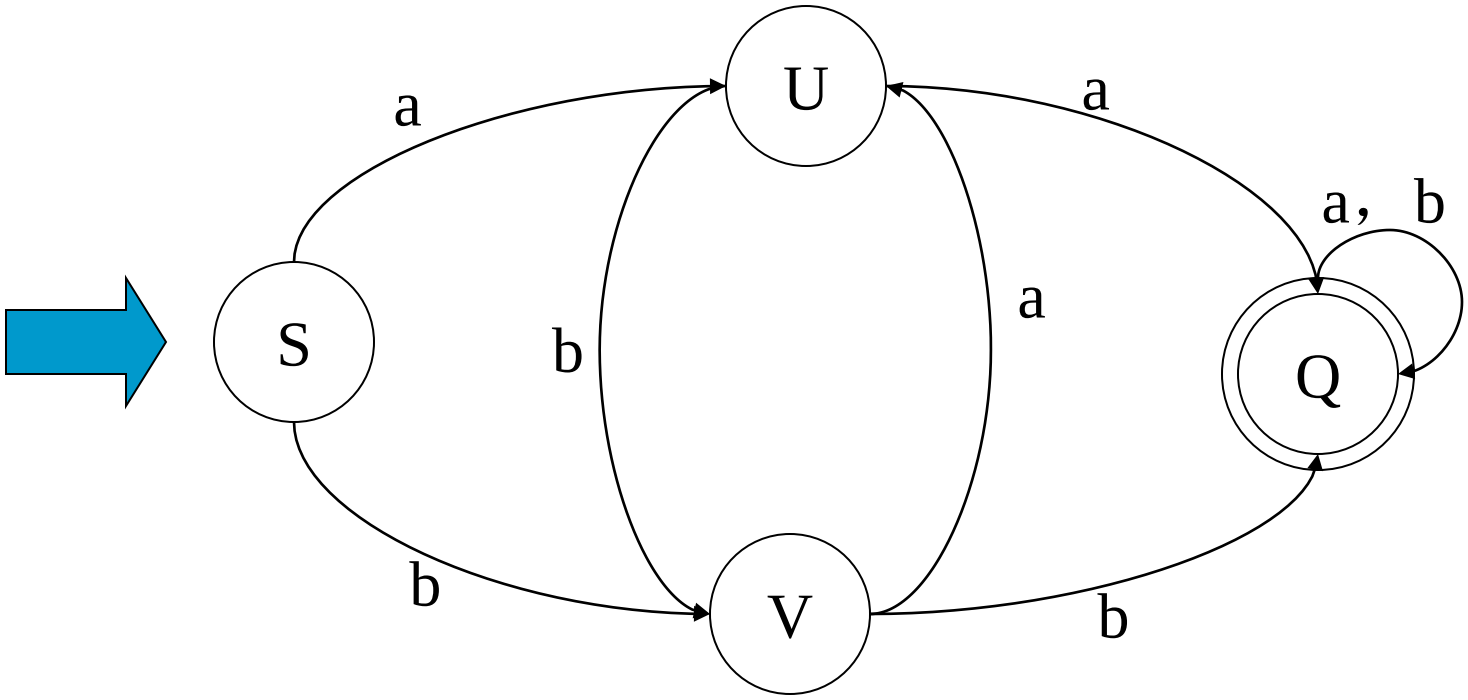
DFA定义

- 3、f是转换函数，是在$K\times\Sigma\rightarrow K$上的映射，即，如$f(k_i, a)=k_j$，（$k_i\in K$，$k_j\in K$）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；
- 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。

DFA 例：

- 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

DFA 的状态转换图表示



DFA 的矩阵表示

状态 \ 字符	a	b
S	U	V
U	Q	V
V	U	Q
Q	Q	Q

DFA

- DFA M的作用：

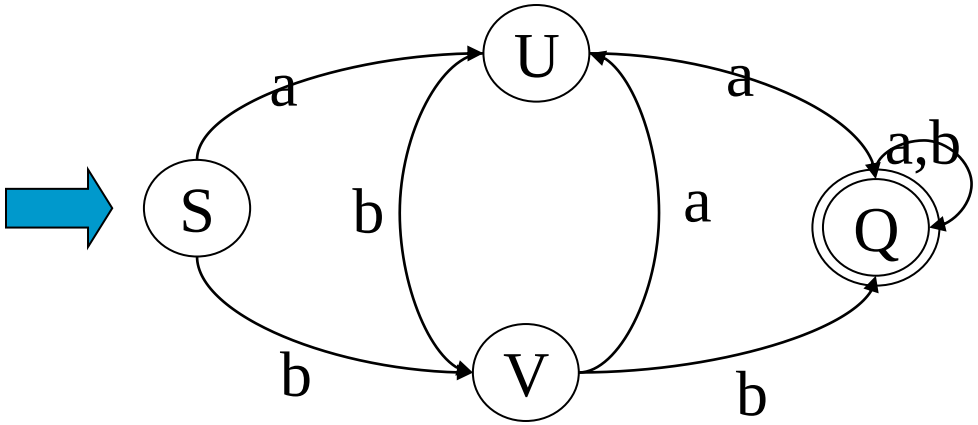
  对于$\sum^*$中的任何字符串t，若存在一条从初态结到某一终态结的道路，且这条路上所有弧的标记符连接成的字符串等于t，则称t可为DFA M所接受（识别）。若M的初态结同时又是终态结，则空字可为M所识别（接受）。

DFA

- $\sum^*$上的符号串t被M接受的形式叙述：
  - 若$t \in \sum^*$，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**）。
- $\sum^*$上的符号串t在M上运行的定义：
  - 一个输入符号串t，（我们将它表示成$t_1\ t_x$的形式，其中$t_1 \in \sum$，$t_x \in \sum^*$）在DFA M上**运行**的定义为：
  - $f(Q,\ t_1\ t_x) = f(f(Q,\ t_1),\ t_x)$ 其中$Q \in K$

DFA

- 例：证明t=baab被前例中的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。

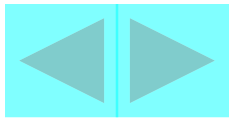


所识别的语言：含有相继两个a或相继两个b的串。

DFA

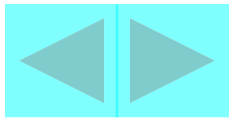
- DFA M所能接受的符号串的全体记为L(M)(语言)
- 结论：
  - ∑上一个符号串集V⊂∑*是正规的，当且仅当存在一个∑上的确定有穷自动机M，使得V=L(M)。

- DFA 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’)



4.3.2不确定的有穷自动机NFA

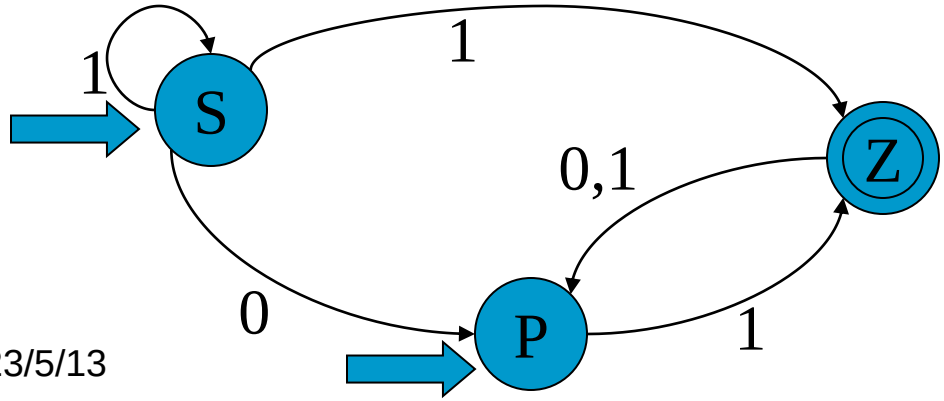
- 定义
  - N=｛K，∑，f，S，Z｝，其中K为状态的有穷非空集，∑为有穷输入字母表，f为K×∑*到K的子集（2^K）的一种映射，S⊂K是初始状态集，Z⊂K为终止状态集。
- 例子
  - NFA N=（{S，P，Z}，{0，1}，f，{S，P}，{Z}）
  - 其中 f（S，0）={P}



4.3.2 NFA

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

- 状态图表示



4.3.2 NFA

- 例：一个NFA M=（{0,1,2,3,4},{a,b},f,{0},{2,4}）其中

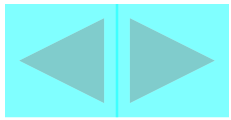
  f(0,a)={0,3}　　f(0,b)={0,1}

  f(1,b)={2}　　f(2,a)={2}

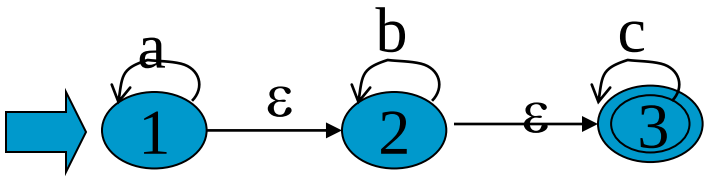
  f(2,b)={2}　　f(3,a)={4}

  f(4,a)={4}　　f(4,b)={4}

- 画出其状态图

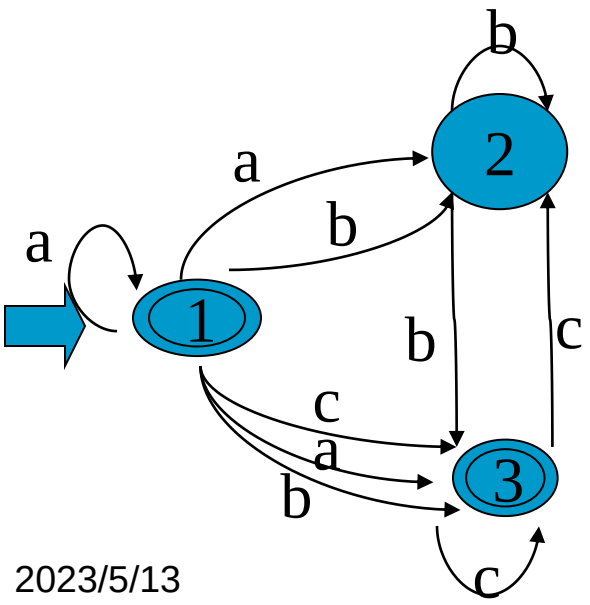


4.3.2 NFA

- $\sum^*$上的符号串t在NFA N上运行...
- $\sum^*$上的符号串t被NFA N接受...
- 具有ε转移的不确定的有穷自动机NFA... f为 K×（$\sum \cup \{\varepsilon\}$）到K的子集（$2^K$）的一种映射
  - 

4.3.2 NFA

– 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)。



4.3.3 NFA的确定化

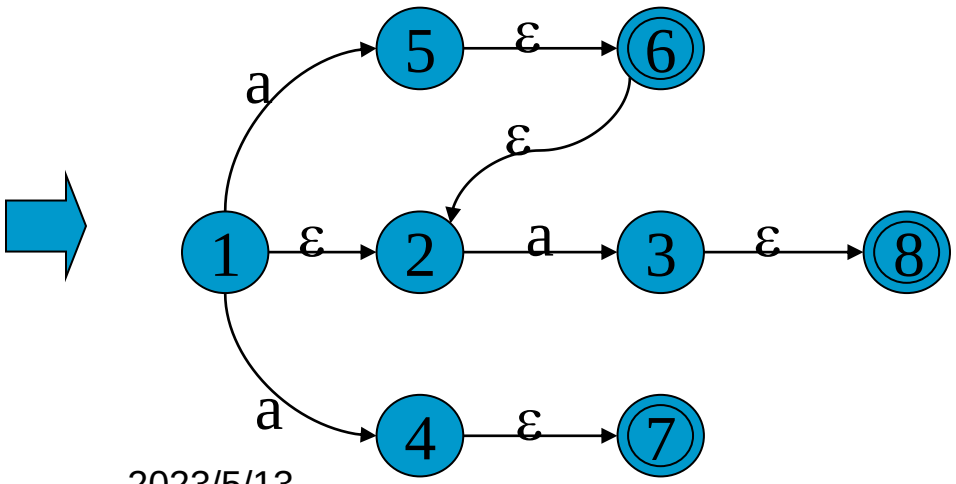
- DFA是NFA的特例。对每个NFA N一定存在一个DFA M，使得L(M)=L(N)。对每个NFA N存在着与之等价的DFA M。与某一NFA等价的DFA不唯一

4.3.3 NFA的确定化

- 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的某一状态经过一条a弧而到达的状态的全体。定义Ia = ε-closure(J)

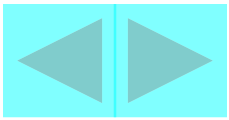
4.3.3 NFA的确定化

- 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};



4.3.3 NFA的确定化

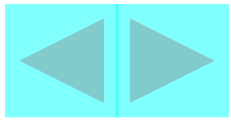
- 假设NFA N=(K, ∑,f,K_0,K_t)按如下办法构造一个DFA M=(S, ∑,d,S_0,S_t)，使得L(M)=L(N)：
  - 1 M的状态集S由K的一些子集组成。用[S_1 S_2... S_j]表示S的元素，其中S_1, S_2,... S_j是K的状态。并且约定，状态S_1, S_2,... S_j是按某种规则排列的，即对于子集{S_1, S_2}={ S_2, S_1,}来说，S的状态就是[S_1 S_2];



4.3.3 NFA的确定化

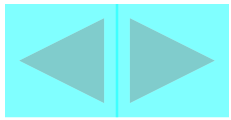
- 2 M和N的输入字母表是相同的，即是Σ;
- 3 转换函数是这样定义的：

  $d([S_1\ S_2,\ldots S_j],a)= [R_1R_2\ldots R_t]$ 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$\{S_i, S_k,\ldots S_e\}\cap K_t\neq\Phi\}$



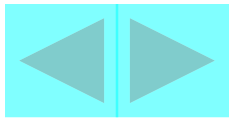
4.3.3 NFA的确定化

- 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_l),其中T_1, T_2,... T_l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。



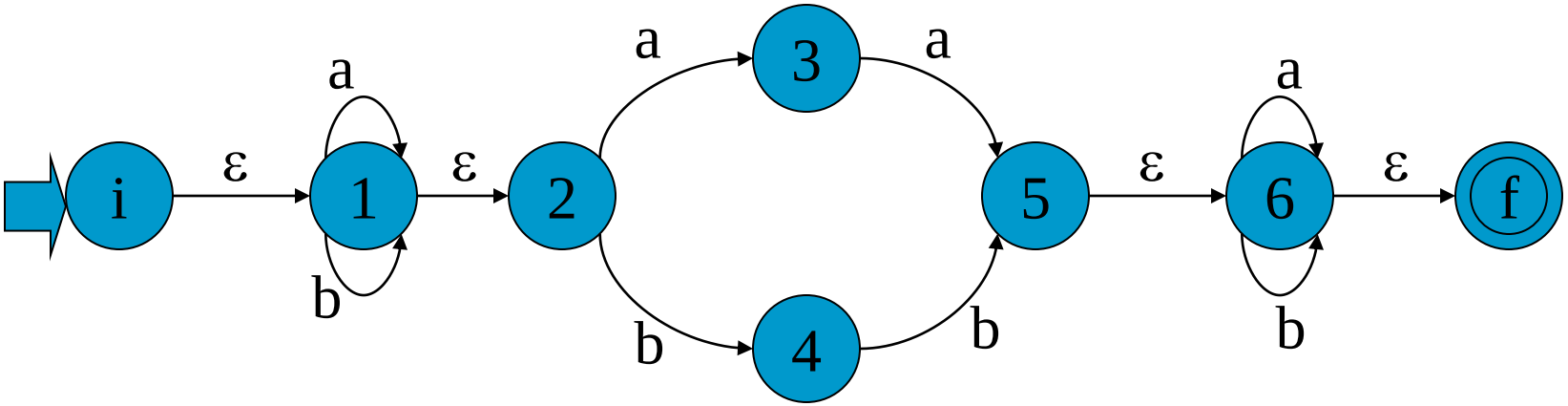
4.3.3 NFA的确定化

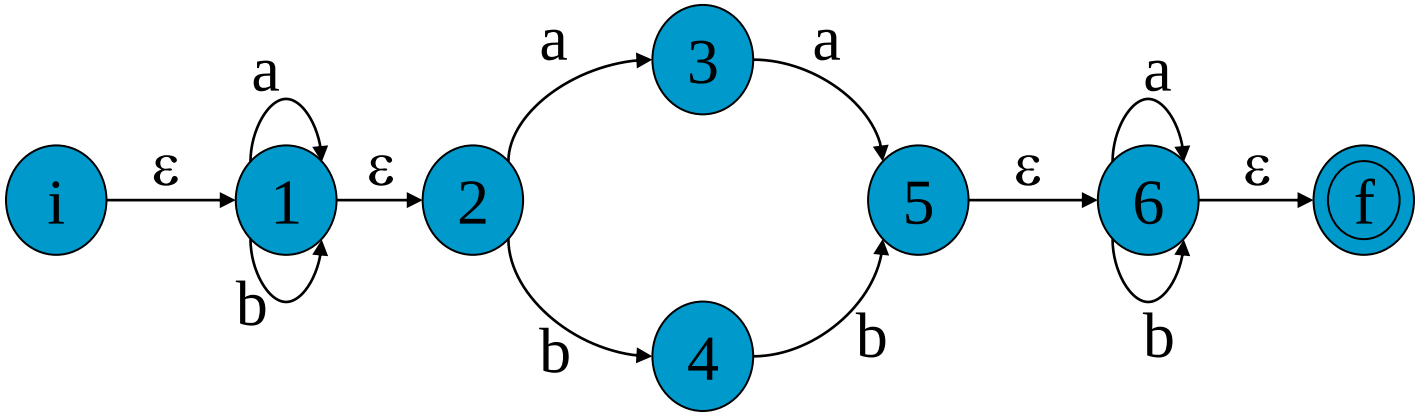
– 2 while （C中存在尚未被标记的子集T）
do {
标记T;
for 每个输入字母a do
{
U:= ε-closure(move(T,a))；
if U不在C中 then
将U作为未标记的子集加
在C中 }
}



4.3.3 NFA的确定化

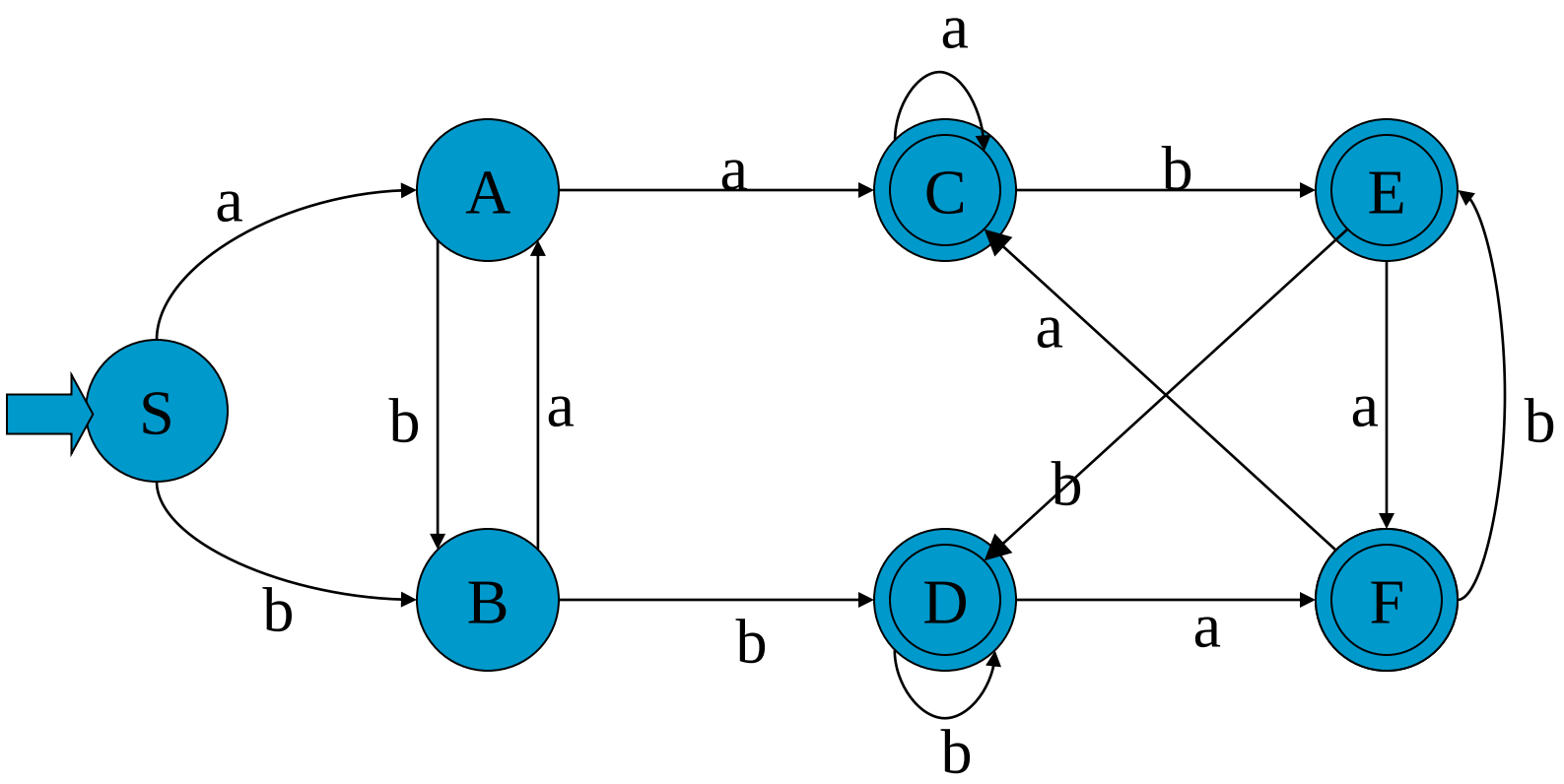
- 例子





	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

4.3.3 NFA的确定化



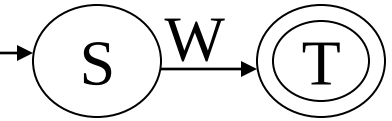
4.4有穷自动机和正规表达式

有穷自动机和正规表达式的等价性：

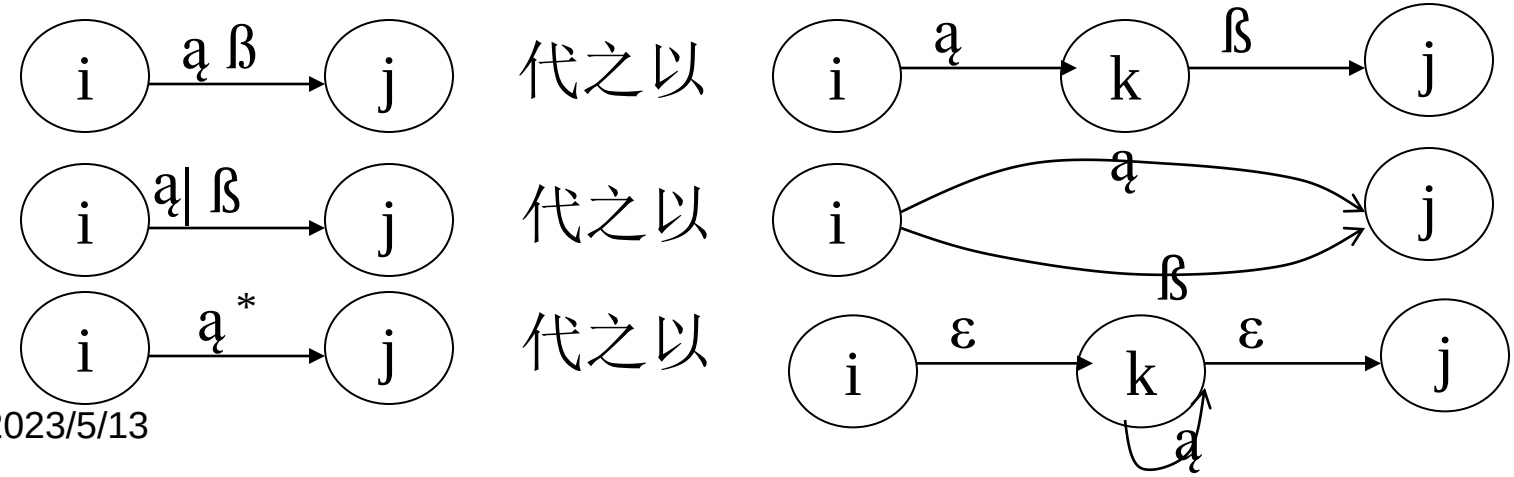
- 1.对于Σ上的一个NFA M，可以构造一个Σ上的正规式R, 使得L(R)=L(M)。
- 2. 对于Σ上的一个正规式R，可以构造一个Σ上的NFA M，使的L(M)=L(R)。

正规式 =>有穷自动机

- 设给定正规式W，则构造相应自动机的方法如下：
- 若W=ø，则对应的NFA为 

- 若W<>ø，则对应的NFA为 

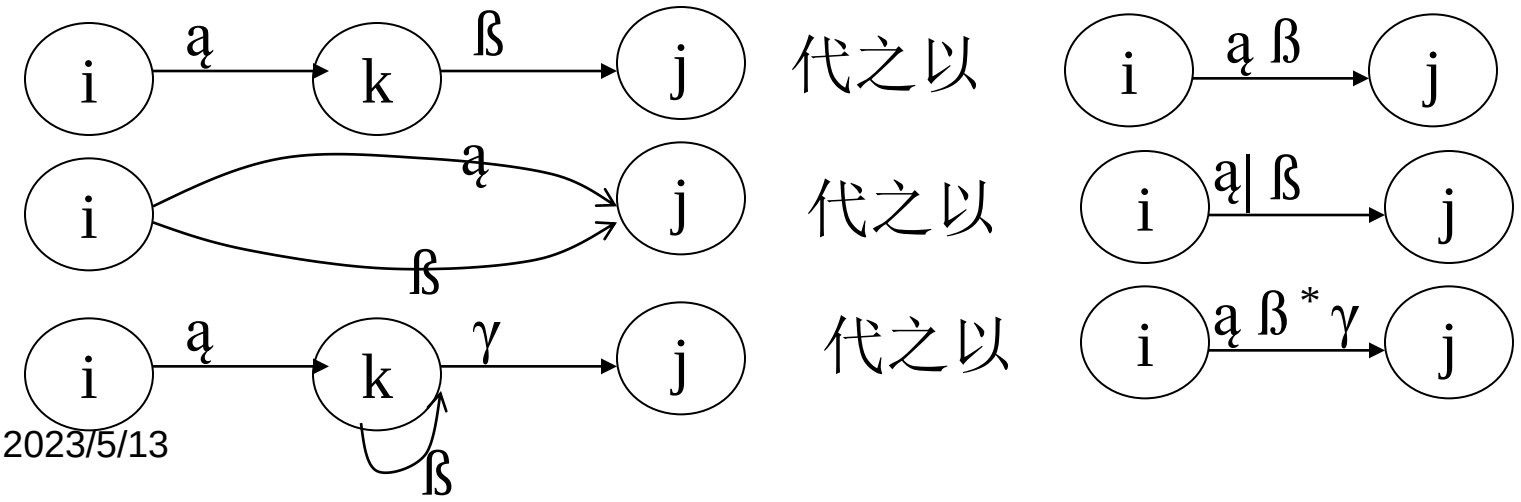
- 然后利用以下规则加入结点和箭弧，直到得到自动机为止。



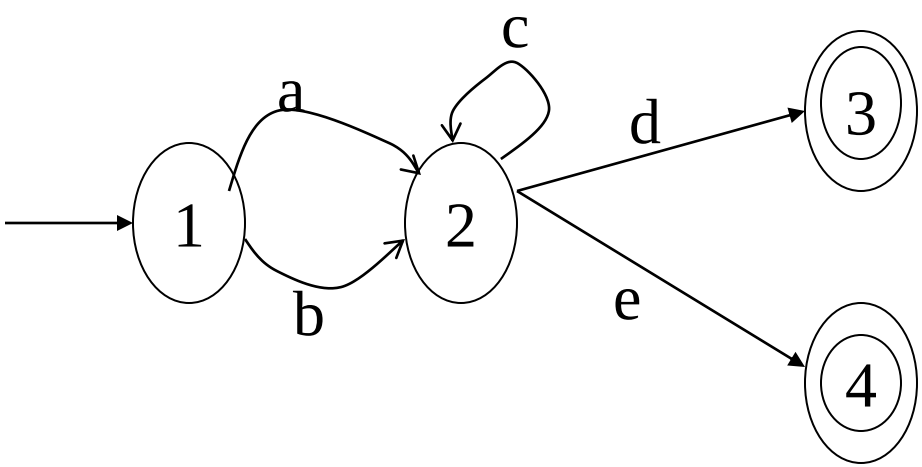
- 例：正规式$(a|b)^*(aa|bb)(a|b)^*$转化为自动机

有穷自动机=>正规式

- 首先检查是否只有一个终态结点，若有多个，则引入新结点T，从所有终态结点引ε边到T结点，并令T为唯一的终态结点。
- 然后按以下规则消除结点与箭弧：

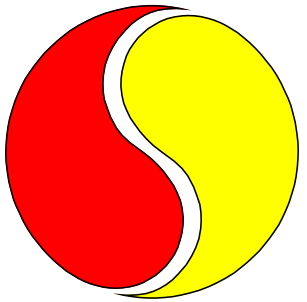


- 例：



4.5有穷自动机和正规文法

- 有穷自动机和正规文法的等价性：
  1.对于一个NFA M，都存在 一个正规文法G，使得L(G)=L(M).
  2.对于一个正规文法G,都存在一个NFA M，使得L(M)=L(G).



正规文法=>自动机

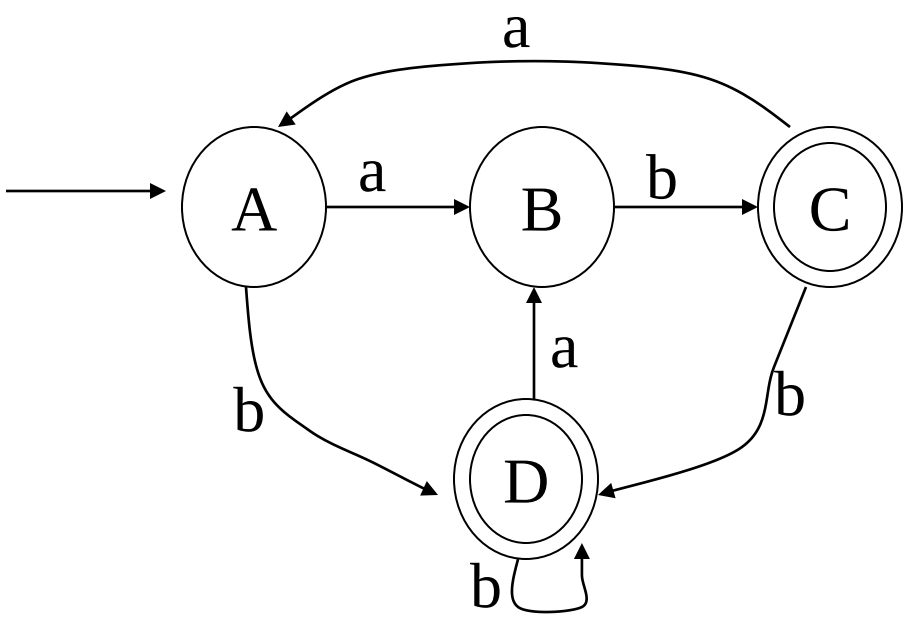
- 字母表与G的终结符集相同；
- G中的非终结符对应状态，开始符对应开始状态
- 增加一个新的终结状态Z。
- G中的A->tB构造转换函数f(A,t)=B
- G中的A->t构造转换函数f(A,t)=Z

- 例：求与G[S]等价的NFA：
- G[S]：
- S->aA
- S->bB
- S->ε
- A->aB
- A->bA
- B->aS
- B-> bA
- B->ε

自动机=>正规文法

- 对转换函数f(A,t)=B，对应产生式：A->tB
- 对终态Z，增加一产生式： Z->ε
- NFA的初态对应文法的开始符号；
- NFA的字母表对应文法的终结符号集。

- 例：给出与下图NFA等价的正规文法G。



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

总结

- 词法分析程序
- 正规表达式与正规集（正规语言）
- 有穷自动机
- 有穷自动机和正规表达式
- 有穷自动机和正规文法