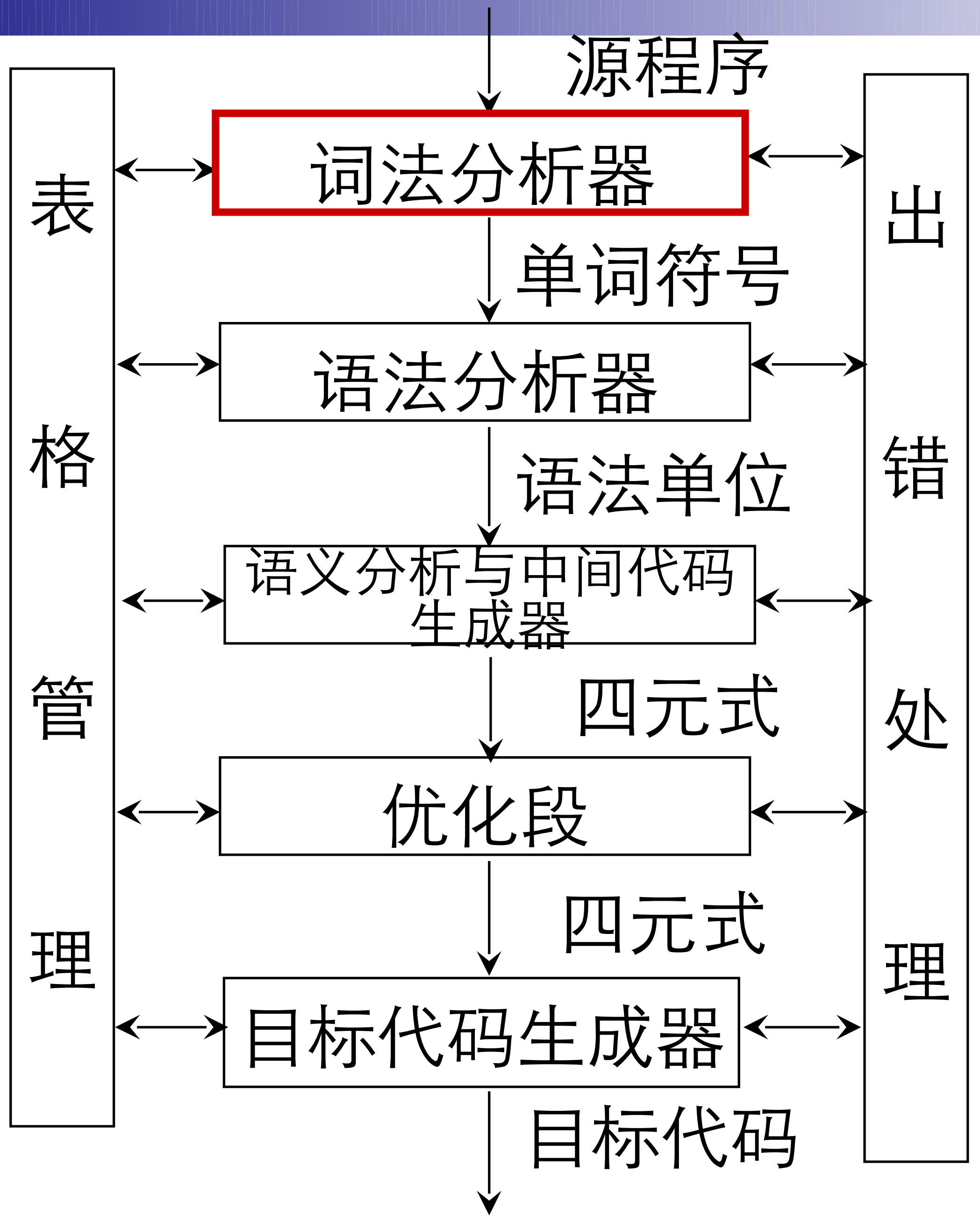
编译原理

第三章 词法分析

编译程序总框



源程序
表格管理
词法分析器
单词符号
语法分析器
语法单位
语义分析与中间代码生成器
四元式
优化段
四元式
目标代码生成器
目标代码
出错处理

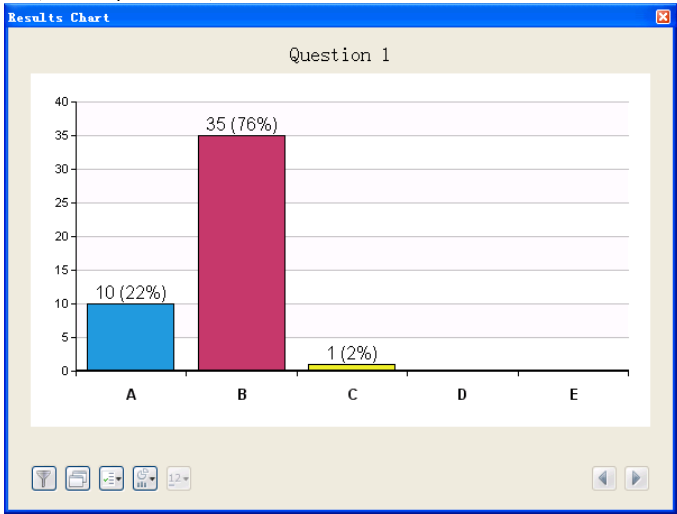
调查：词法分析程序

在操作系统的“ shell 命令解释器”实验中，你是如何设计和实现命令的单词识别程序的（ ）

A. 全部自己实现

B. 使用 LEX(FLEX) 工具实现

C. 使用其它词法分析程序开发工具实现



iclicker

第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

第三章 词法分析

- 词法分析的任务
  - 从左至右逐个字符地对源程序进行扫描，产生一个个单词符号
- 词法分析器 (Lexical Analyzer) 又称扫描器(Scanner)
  - 执行词法分析的程序

3.1 对于词法分析器的要求

- 功能
  - 输入源程序、输出单词符号
- 单词符号的种类
  - 基本字：如 begin， repeat， …
  - 标识符——表示各种名字：如变量名、数组名和过程名
  - 常数：各种类型的常数
  - 运算符：+， -， *， /， …
  - 界符：逗号、分号、括号和空白

- 输出的单词符号的表示形式
  - （单词种别，单词自身的值）
- 单词种别通常用整数编码表示
  - 若一个种别只有一个单词符号，则种别编码就代表该单词符号。假定基本字、运算符和界符都是一符一种。
  - 若一个种别有多个单词符号，则对于每个单词符号，给出种别编码和自身的值。
    - 标识符单列一种；标识符自身的值表示成按机器字节划分的内部码
    - 常数按类型分种；常数的值则表示成标准的二进制形式

例 FORTRAN 程序

- IF (5.EQ.M) GOTO 100
- 输出单词符号
  - 逻辑 IF (34， -)
  - 左括号 (2， -)
  - 整常数 (20， ‘5’ 的二进制)
  - 等号 (6， -)
  - 标识符 (26， ‘M’)
  - 右括号 (16， -)
  - GOTO (30， -)
  - 标号 (19， ‘100’ 的二进制)

例 C 程序

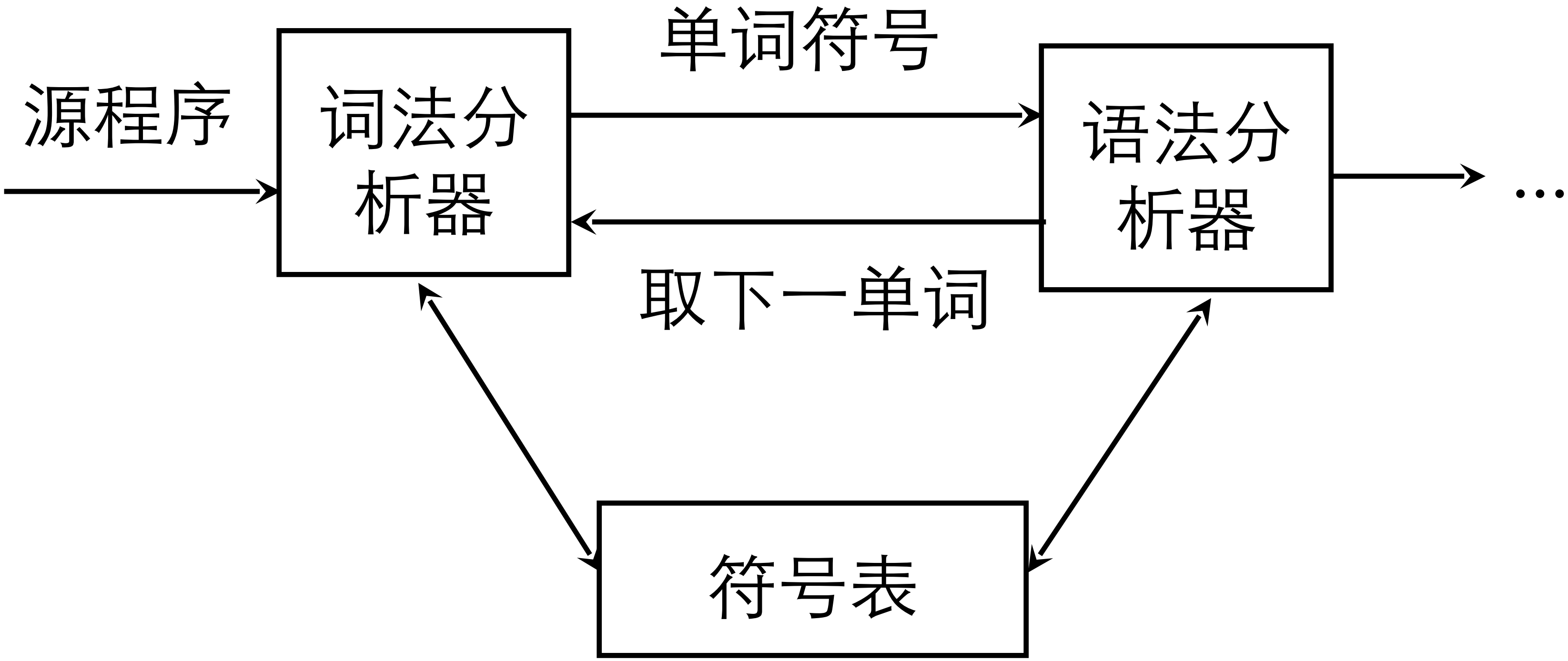
- while (i>=j) i--;
- 输出单词符号
  - < while， - >
  - < (， - >
  - < id， 指向 i 的符号表项的指针 >
  - < >=， - >
  - < id， 指向 j 的符号表项的指针 >
  - <)， - >
  - < id， 指向 i 的符号表项的指针 >
  - < --， - >
  - < ;， - >

词法分析器作为一个独立子程序

- 词法分析是作为一个独立的阶段，是否应当将其处理为一遍呢?
  - 作为独立阶段的优点
    - 结构简洁、清晰和条理化，有利于集中考虑词法分析一些枝节问题
  - 不作为一遍
    - 将其处理为一个子程序

- 计算思维
  - 分解
  - 权衡

词法分析器在编译器中地位



第三章 词法分析

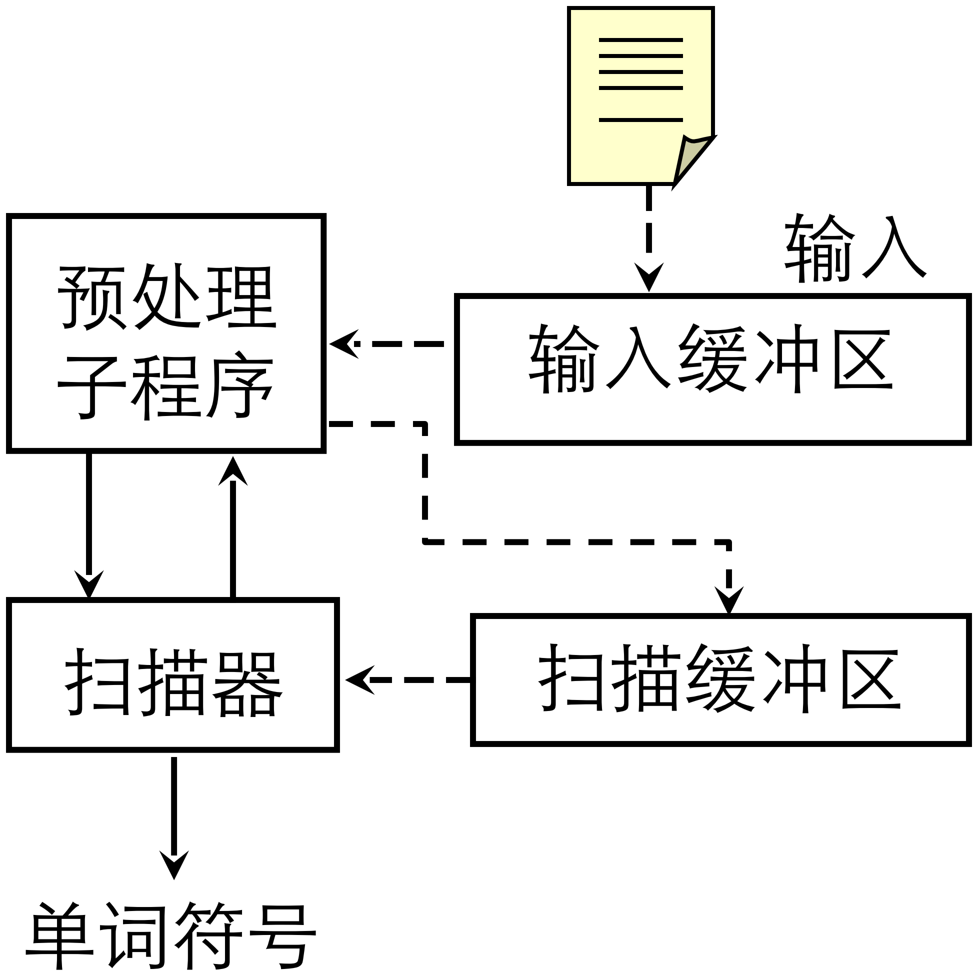
- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

- 计算思维
  - 分解

3.2 词法分析器的设计

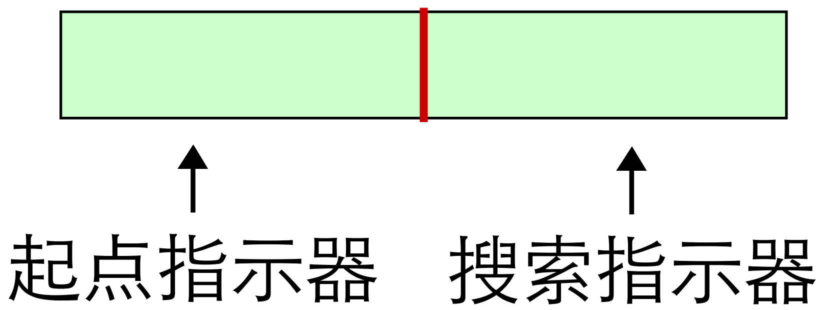
- 词法分析器的结构

•剔除无用的空白、跳格、回车和换行等编辑性字符
•区分标号区、捻接续行和给出句末符等



输入、预处理

- 扫描缓冲区



WhatALong…Word

rd……………………………………………………

… WhatALong…Wo | rd………………………

rd……………………… | … WhatALong…Wo

两个半区互补使用
单词长度限制 = 半区的长度

单词符号的识别：超前搜索

- 基本字识别
- 例如

```
DO99K=1，10          DO 99 K = 1， 10
DO99K=1.10
IF(5.EQ.M)GOTO55 IF (5.EQ.M) GOTO 55
IF(5)=55
```

- 需要超前搜索才能确定哪些是基本字

- 标识符识别
  - 字母开头的字母数字串，后跟界符或算符
- 常数识别
  - 识别出算术常数并将其转变为二进制内码表示。有些也要超前搜索。
    5.EQ.M
    5.E08
- 算符和界符的识别
  - 把多个字符符合而成的算符和界符拼合成一个单一单词符号。
    :=， **， .EQ. ， ++， --， >=

几点限制——不必使用超前搜索

- 所有基本字都是保留字；用户不能用它们作自己的标识符
- 基本字作为特殊的标识符来处理，使用保留字表
- 如果基本字、标识符和常数（或标号）之间没有确定的运算符或界符作间隔，则必须使用一个空白符作间隔

  DO99K=1， 10

  要写成 DO 99 K=1， 10

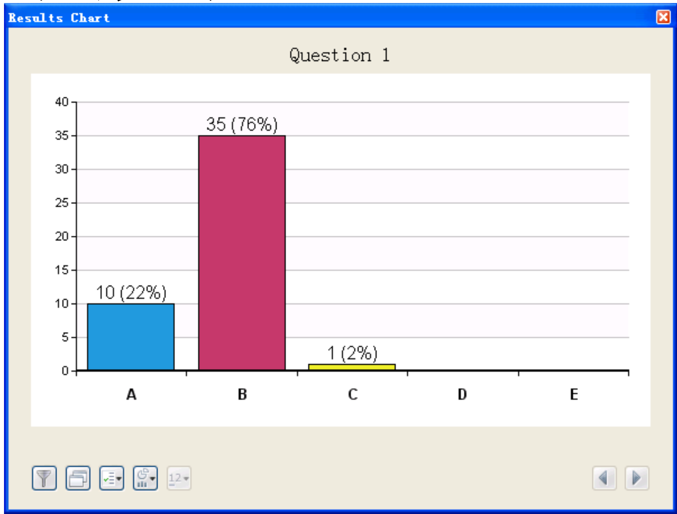
调查：词法分析程序

在操作系统的“shell 命令解释器”实验中，你是如何设计和实现命令的单词识别程序的（　）

A. 全部自己实现

B. 使用 LEX(FLEX) 工具实现

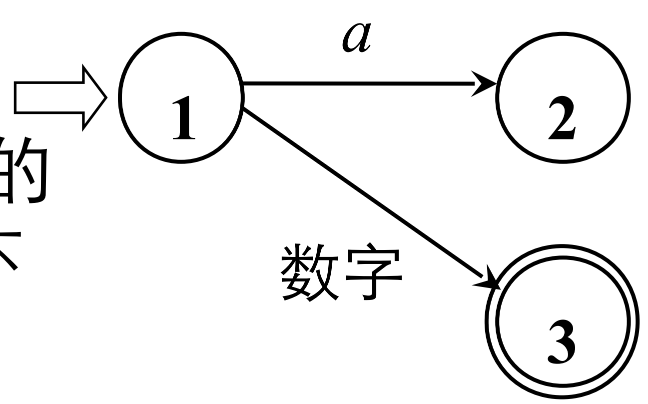
C. 使用其它词法分析程序开发工具实现



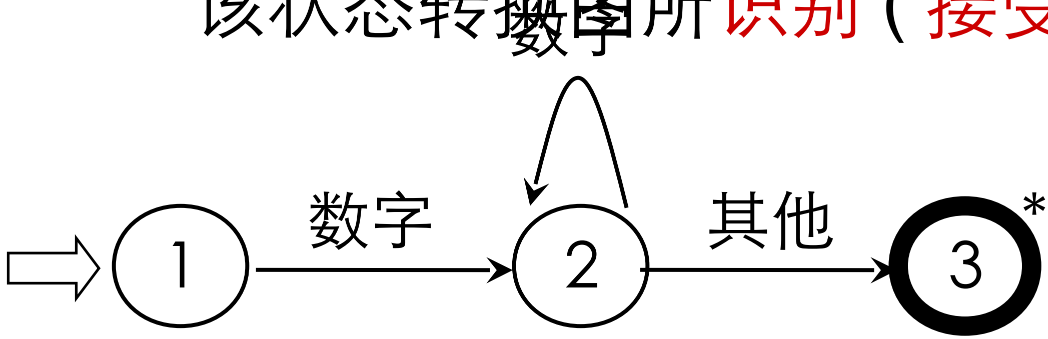
iclicker

状态转换图

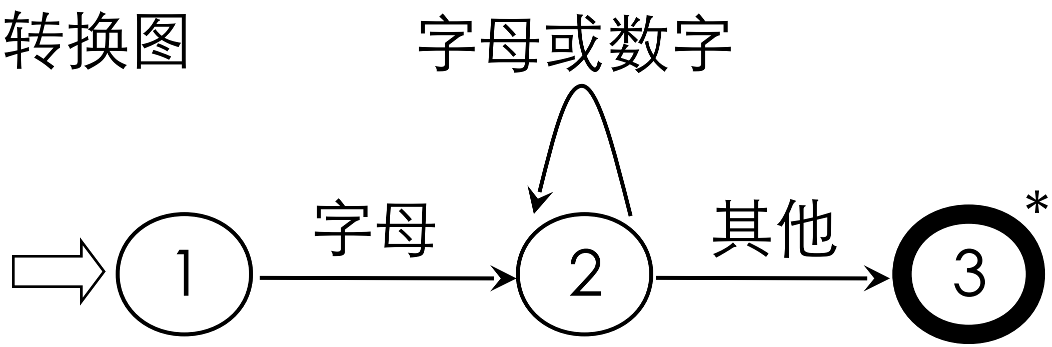
- 状态转换图是一张有限方向图
  - 结点代表状态，用圆圈表示
  - 状态之间用箭弧连结，箭弧上的标记(字符)代表射出结状态下可能出现的输入字符或字符类
  - 一张转换图只包含有限个状态，其中有一个为初态，至少要有一个终态



- 状态转换图可用于识别（或接受）一定的字符串
  - 若存在一条从初态到某一终态的道路，且这条路上所有弧上的标记符连接成的字等于α，则称α为该状态转换图所识别（接受）



识别整常数的状态转换图

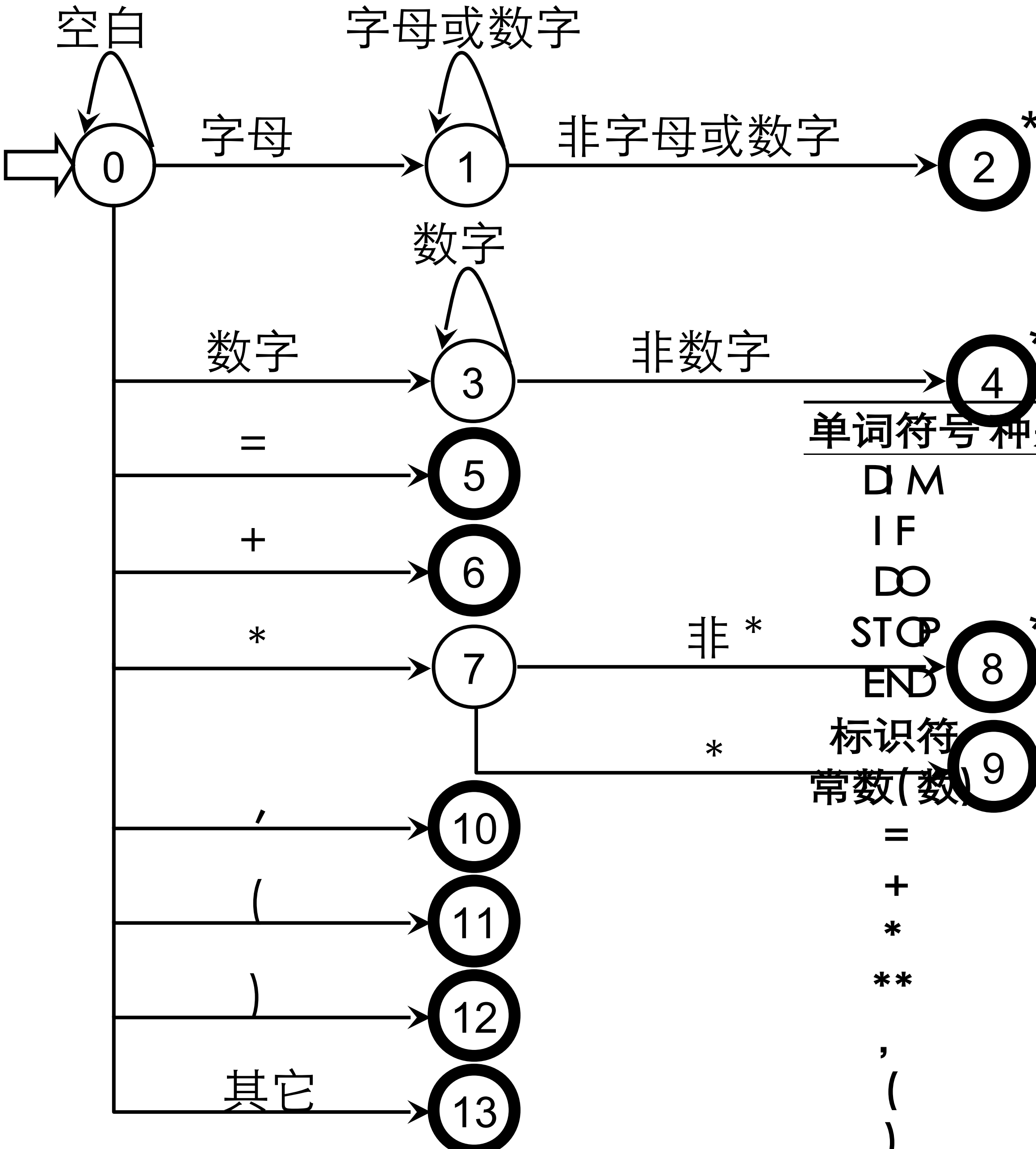


识别标识符的状态转换图

词法分析器的设计示例

- 助忆符
  - 直接用编码表示不便于记忆，因此用助忆符来表示编码。

单词符号	种别编码	助忆符	内码值
DIM	1	$DIM	-
IF	2	$IF	-
DO	3	$DO	-
STOP	4	$STOP	-
END	5	$END	-
标识符	6	$ID	内部字符串
常数（数）	7	$INT	标准二进制形式
=	8	$ASSIGN	-
+	9	$PLUS	-
*	10	$STAR	-
**	11	$POWER	-
,	12	$COMMA	-
(	13	$LPAR	-
)	14	$RPAR	-



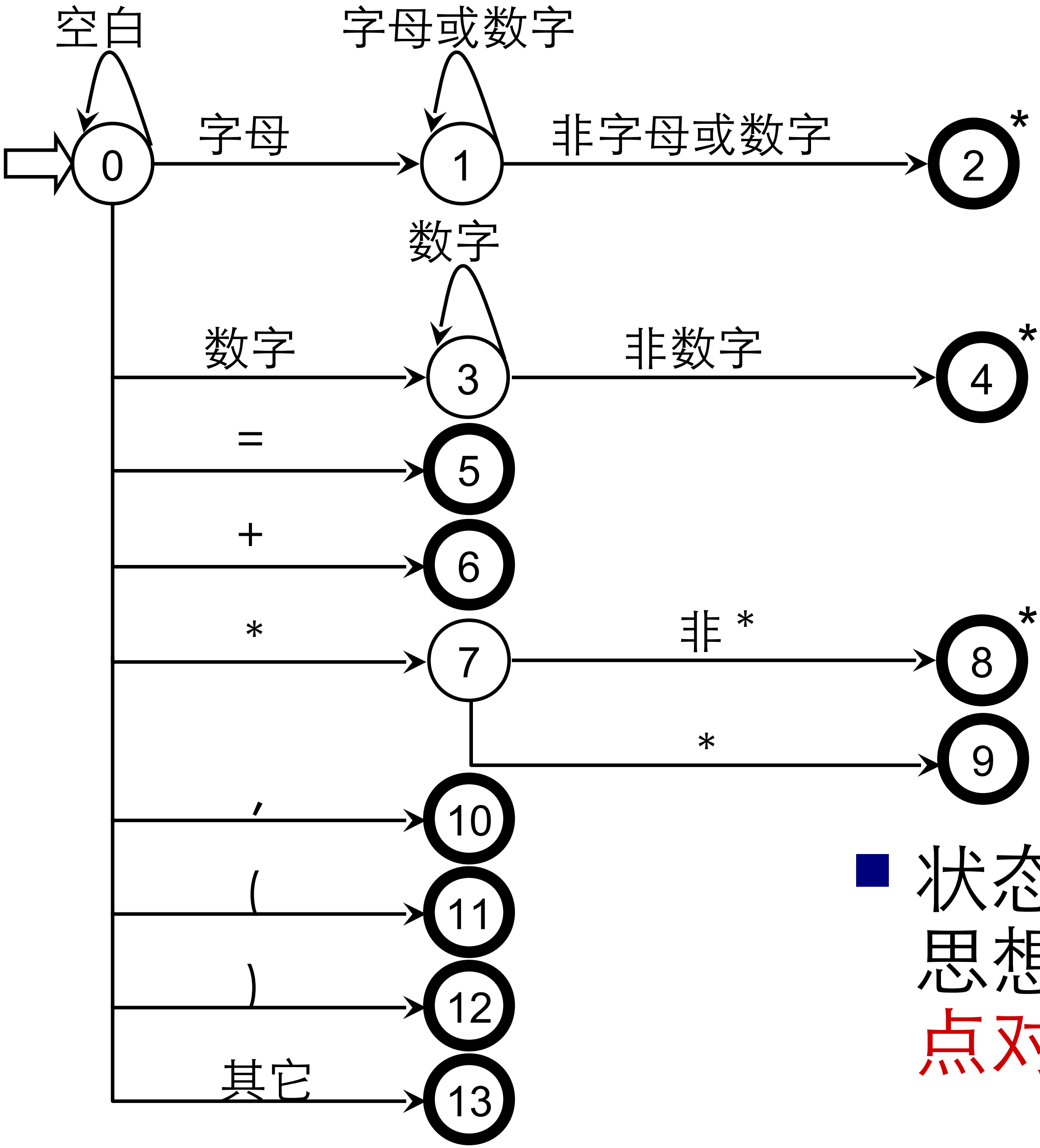
单词符号	种别编码	助忆符	内码值
DIM	1	$DIM	-
IF	2	$IF	-
DO	3	$DO	-
STOP	4	$STOP	-
END	5	$END	-
标识符	6	$ID	内部字符串
常数(整)	7	$INT	标准二进制形式
=	8	$ASSIGN	-
+	9	$PLUS	-
*	10	$STAR	-
**	11	$POWER	-
,	12	$COMMA	-
(	13	$LPAR	-
)	14	$RPAR	-

几点限制——不必使用超前搜索

- 所有基本字都是保留字；用户不能用它们作自己的标识符
- 基本字作为特殊的标识符来处理，使用保留字表
- 如果基本字、标识符和常数（或标号）之间没有确定的运算符或界符作间隔，则必须使用一个空白符作间隔

  DO99K=1， 10

  要写成　DO 99 K=1， 10

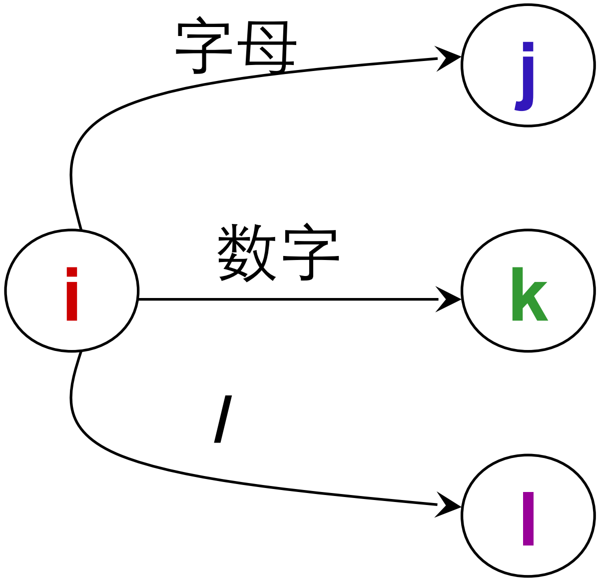


- 状态转换图的实现思想：每个状态结点对应一小段程序

状态转换图的实现

- 思想：每个状态结点对应一小段程序
- 具体方法

  1) 对不含回路的分叉结点，可用一个 CASE 语句或一组 IF-THEN-ELSE 语句实现

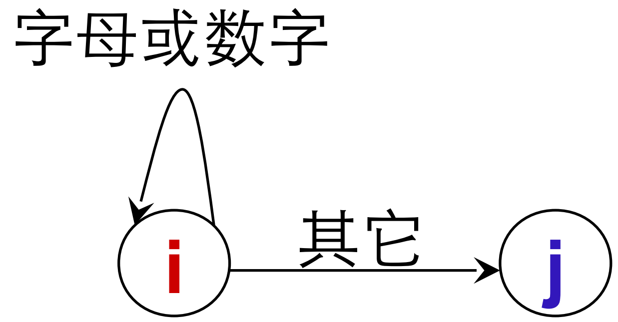



```
GetChar( );
if (IsLetter( ))
    {… 状态 j 的对应程序段… ;}
else if (IsDigit( ))
    {… 状态 k 的对应程序段… ;}
else if (ch='/')
    {… 状态 l 的对应程序段… ;}
else
    {… 错误处理… ;}
```

状态转换图的实现

- 具体方法

  2) 对含回路的状态结点，可对应一段由 WHILE 结构和 IF 语句构成的程序．




```
GetChar( );
while (IsLetter( )or IsDigit( ))
    GetChar( );
··· 状态 j 的对应程序段···
```

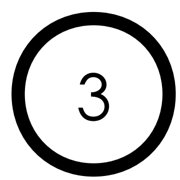
状态转换图的实现

- 具体方法

  3) 终态结点表示识别出某种单词符号，因此，对应语句为

  RETURN (C ， VAL)

  其中， C 为单词种别， VAL 为单词自身值



- 全局变量与过程

1) ch 字符变量、存放最新读入的源程序字符

2) strToken 字符数组，存放构成单词符号的字符串

3) GetChar 子程序过程，把下一个字符读入到 ch 中

4) GetBC 子程序过程，跳过空白符，直至 ch 中读入一非空白符

5) Concat 子程序，把 ch 中的字符连接到 strToken

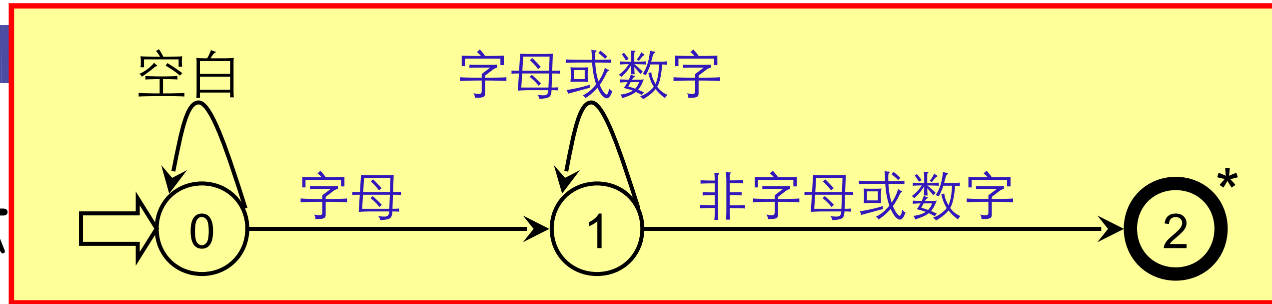
6)IsLetter 和 IsDisgital 布尔函数，判断 ch 中字符是否为字母和数字

7) Reserve 整型函数，对于 strToken 中的字符串查找保留字表，若它实保留字则给出它的编码，否则回送 0

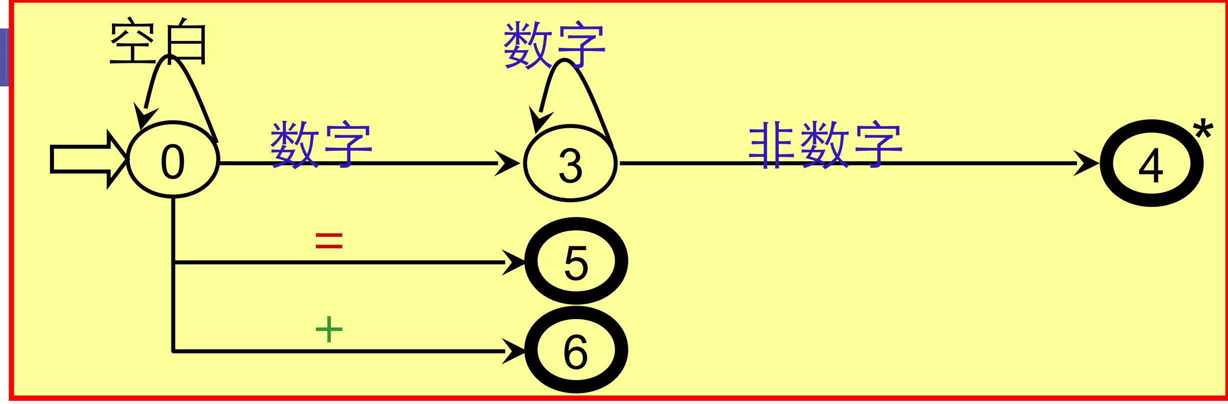
8) Retract 子程序，把搜索指针回调一个字符位置

9)InsertId 整型函数，将 strToken 中的标识符插入符号表，返回符号表指针

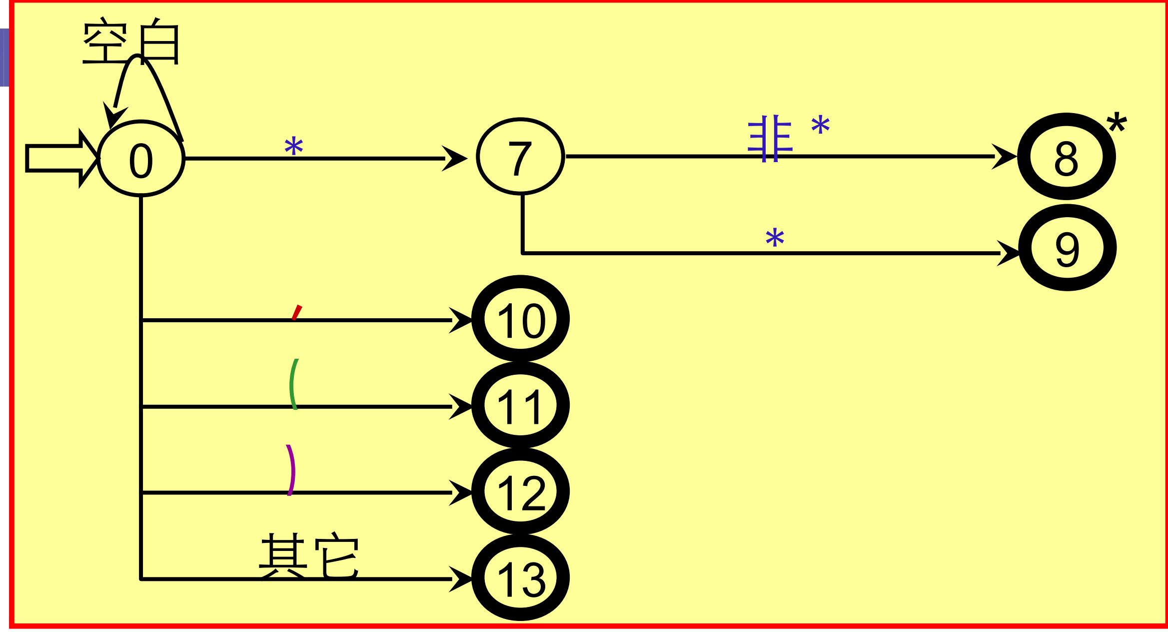
10)InsertConst 整型函数过程，将 strToken 中的常数插入常数表，返回常数表指针。




```
int code, val
strToken := “
GetChar(); GetBC();
if (IsLetter())
begin
      while (IsLetter() or IsDigit())
      begin
            Concat(); GetChar();
      end
      Retract();
      code := Reserve();
      if (code = 0)
      begin
            value := InsertId(strToken);
            return ($ID, value);
      end
      else
            return (code, -);
end
```

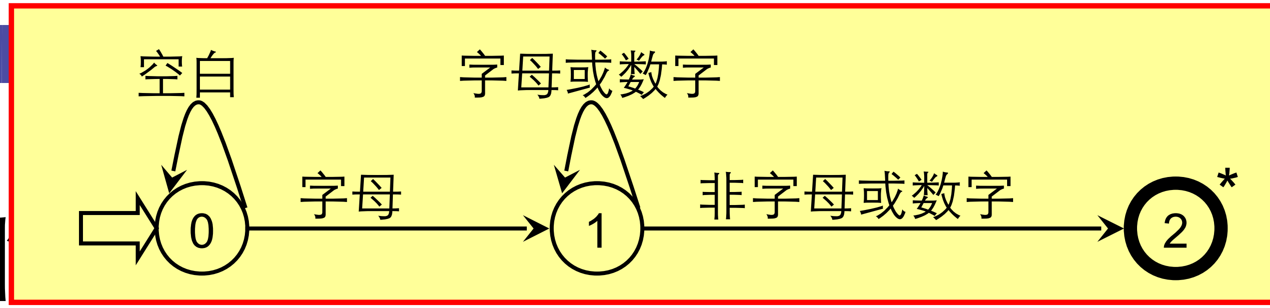



```
else if (IsDigit())
begin
     while (IsDigit())
     begin
          Concat( ); GetChar( );
     end
     Retract();
     value := InsertConst(strToken);
     return($INT, value);
end
else if (ch ='=') return ($ASSIGN, -);
else if (ch ='+') return ($PLUS, -);
```




```
else if (ch ='*')
begin
      GetChar();
      if (ch ='*') return ($POWER, -);
      Retract(); return ($STAR, -);
end
else if (ch =',') return ($COMMA, -);
else if (ch ='(') return ($LPAR, -);
else if (ch =')') return ($RPAR, -);
else ProcError( );        /* 错误处理 */
```

将状态图的



- 变量 curState 用于保存现有的状态
- 用二维数组表示状态图： stateTrans[state][char]

```
curState = 初态
GetChar();
while( stateTrans[curState][c
   // 存在后继状态，读入、拼接
   Concat();
   // 转换入下一状态，读入下一字符
  curState= stateTrans[curState][ch];
   if cur_state 是终态 then 返回 strToken 中的单词
   GetChar( );
}
```

只是个框架，还有很多细节需要考虑！是否有自动的方法产生词法分析程序？

小结

- 词法分析器的功能
- 词法分析器的设计
    - 状态转换图
    - 状态转换图的实现

作业

- 阅读： PL 语言编译器的词法分析子程序getsym
- 思考：如果语言扩展需要增加关键字或新数据类型的常量，需要如何修改程序?

编译原理

第三章 词法分析

第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

回顾

- 词法分析器的功能
- 词法分析器的设计
  - 状态转换图
  - 状态转换图的实现

是否有自动的方法产生词法分析程序？

第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

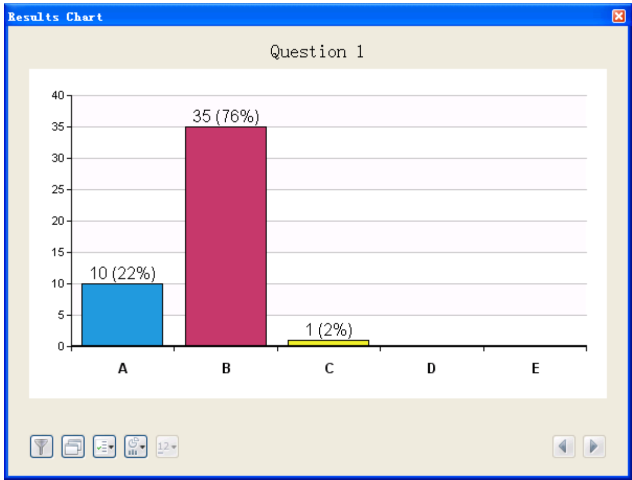
调查：词法分析程序

在操作系统的“ shell 命令解释器”实验中，你是如何设计和实现命令的单词识别程序的 (   )

A. 全部自己实现

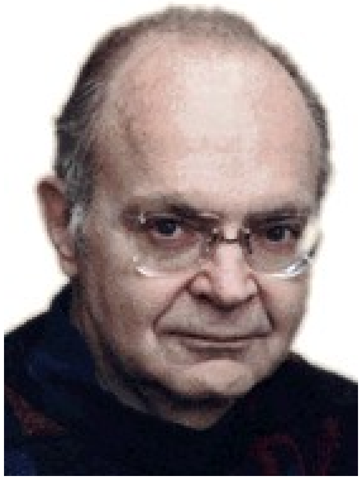
B. 使用 LEX(FLEX) 工具实现

C. 使用其它词法分析程序开发工具实现



iclicker.

Knuth on *Theory and Practice*



Donald Ervin Knuth

Theory and practice are not mutually exclusive; they are intimately connected. They live together and support each other.

3.3 正规表达式与有限自动机

- 几个概念
  - 考虑一个有穷 字母表∑ 字符集
  - 其中每一个元素称为一个字符
  - ∑ 上的字(也叫字符串) 是指由∑中的字符所构成的一个有穷序列
  - 不包含任何字符的序列称为空字，记为 ε
  - 用∑ * 表示∑上的所有字的全体，包含空字 ε
  - 例如：设 ∑ ={a， b}，则

    $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, ...\}$

- Σ^* 的子集 U 和 V 的连接（积）定义为

$$UV = \{ \alpha\beta \mid \alpha \in U \ \& \ \beta \in V \}$$

- V 自身的 n 次积记为

$$V^n = V\,V \cdots V$$

- 规定 $V^0=\{\varepsilon\}$
- 令

$$V^* = V^0 \cup V^1 \cup V^2 \cup V^3 \cup \cdots$$

称 V^* 是 V 的闭包

- 记 $V^+ = V\,V^*$，称 V^+ 是 V 的正规闭包

3.3.1 正规式和正规集

- 正规集可以用正规表达式（简称正规式）表示
- 正规表达式是表示正规集一种方法
- 一个字集合是正规集当且仅当它能用正规式表示

冯 - 诺伊曼构造自然数的方案

- $\varnothing$ 0
- $\{\varnothing\}$ 1
- $\{\varnothing, \{\varnothing\}\}$ 2
- $\{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}$ 3

正规式和正规集的递归定义

- 对给定的字母表Σ

  1)ε 和∅都是Σ上的正规式，它们所表示的正规集为 {ε} 和∅；

  2) 任何 a∈Σ ， a 是Σ上的正规式，它所表示的正规集为 {a}；

正规式和正规集的递归定义（续）

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为 $L(e_1)$ 和 $L(e_2)$ ，则

i） $(e_1 | e_2)$ 为正规式，它所表示的正规集为 $L(e_1) \cup L(e_2)$

ii） $(e_1.e_2)$ 为正规式，它所表示的正规集为 $L(e_1)L(e_2)$

iii） $(e_1)^*$ 为正规式，它所表示的正规集为 $(L(e_1))^*$

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，仅由这些正规式表示的字集才是Σ上的正规集。

- 所有词法结构一般都可以用正规式描述
- 若两个正规式所表示的正规集相同，则称这两个正规式**等价**。如

$$b(ab)^*=(ba)^*b$$

$L(b(ab)^*)$
$= L(b)L((ab)^*)$
$= L(b)\ (L(ab))^*$
$= L(b)\ (L(a)L(b))^*$
$=\{b\}\ \{ab\}^*$
$= \{b\}\ \{\varepsilon, ab, abab, ababab, \ldots\}$
$= \{b, bab, babab, bababab, \ldots\}$

请证明：

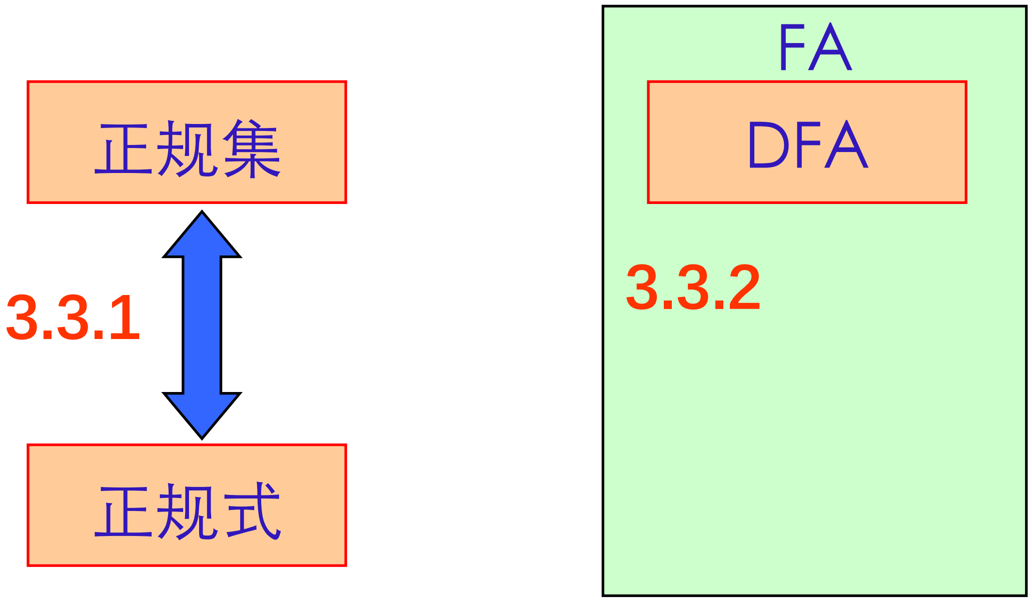
$$(a^*b^*)^*=(a|b)^*$$

$L(\ (ba)^*b)$
(b)
b)
$^*\ L(b)$
$= \{ba\}^*\ \{b\}$
$= \{\varepsilon, ba, baba, bababa, \ldots\}\ \{b\}$
$= \{b, bab, babab, bababab, \ldots\}$

$\because L(b(ab)^*)= L(\ (ba)^*b)$ $\quad\therefore b(ab)^*=(ba)^*b$

- 对正规式，下列等价成立：
  - $e_1 \mid e_2 = e_2 \mid e_1$ 交换律
  - $e_1 \mid (e_2 \mid e_3) = (e_1 \mid e_2) \mid e_3$ 结合律
  - $e_1 (e_2 e_3) = (e_1 e_2) e_3$ 结合律
  - $e_1 (e_2 \mid e_3) = e_1 e_2 \mid e_1 e_3$ 分配律
  - $(e_2 \mid e_3) e_1 = e_2 e_1 \mid e_3 e_1$ 分配律
  - $e\varepsilon = \varepsilon e = e$ $\quad e_1 e_2 <> e_2 e_1$

$L(e_1|e_2)$
$= L(e_1) \cup L(e_2)$
$= L(e_2) \cup L(e_1)$
$= L(e_2|e_1)$


FA
DFA
3.3.2
正规集
3.3.1
正规式

Deterministic Finite Automata

3.3.2 确定有限自动机 (DFA)

- 对状态图进行形式化，则可以下定义:

确定有限自动机 (DFA)M 是一个五元式 $M=(S, \Sigma, f, S_0, F)$, 其中:

1. S: 有穷状态集
2. Σ : 输入字母表 (有穷)
3. f: 状态转换函数，为 $S\times\Sigma\rightarrow S$ 的单值部分映射，$f(s, a)=s'$ 表示: 当现行状态为 s，输入字符为 a 时，将状态转换到下一状态 s'，s' 称为 s 的一个后继状态
4. $S_0\in S$ 是唯一的一个初态
5 $F\subseteq S$: 终态集 (可空)

- 例如： DFA M=({0， 1， 2， 3}， {a， b}， f， 0， {3})， 其中： f定义如下：

  f(0， a)=1　　　f(0， b)=2

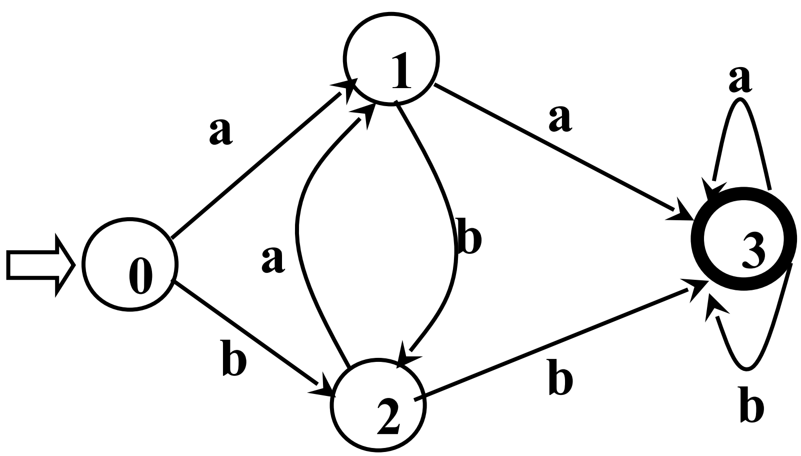
  f(1， a)=3　　　f(1， b)=2

  f(2， a)=1　　　f(2， b)=3

  f(3， a)=3　　　f(3， b)=3

	a	b
0	1	2
1	3	2
2	1	3
3	3	3

状态转换矩阵

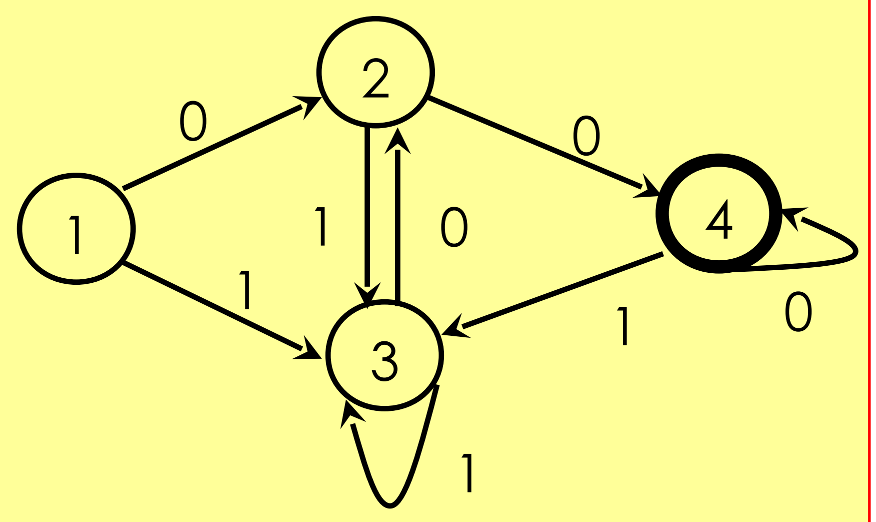


状态转换图

- DFA 可以表示为状态转换图
  - 假定 DFA M 含有 m 个状态和 n 个输入字符
  - 这个图含有 m 个状态结点，每个结点**顶多**含有 n 条箭弧射出，且每条箭弧用 Σ 上的不同的输入字符来作标记

- 对于Σ＊中的任何字α，若存在一条从初态到某一终态的道路，且这条路上所有弧上的标记符连接成的字等于α，则称α为 DFA M 所识别 (接收)
- DFA M 所识别的字的全体记为 L(M)

L(M)={ 以 00 结尾的串 }



DFA M

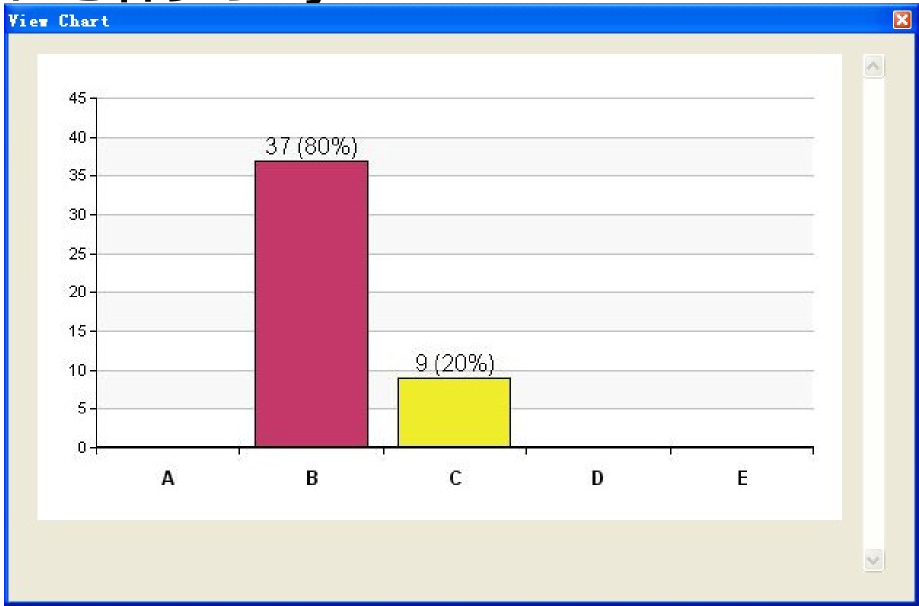
练习

- 图中 DFA M 识别的 L(M) 是什么?

A. L(M)={ 以 aa 或 bb 开头的字 }

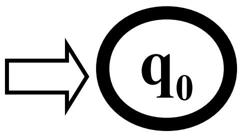
B. L(M)={ 含 aa 或 bb 的字 }

C. L(M)={ 以 aa 或 bb 结尾的字 }

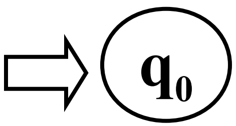


练习

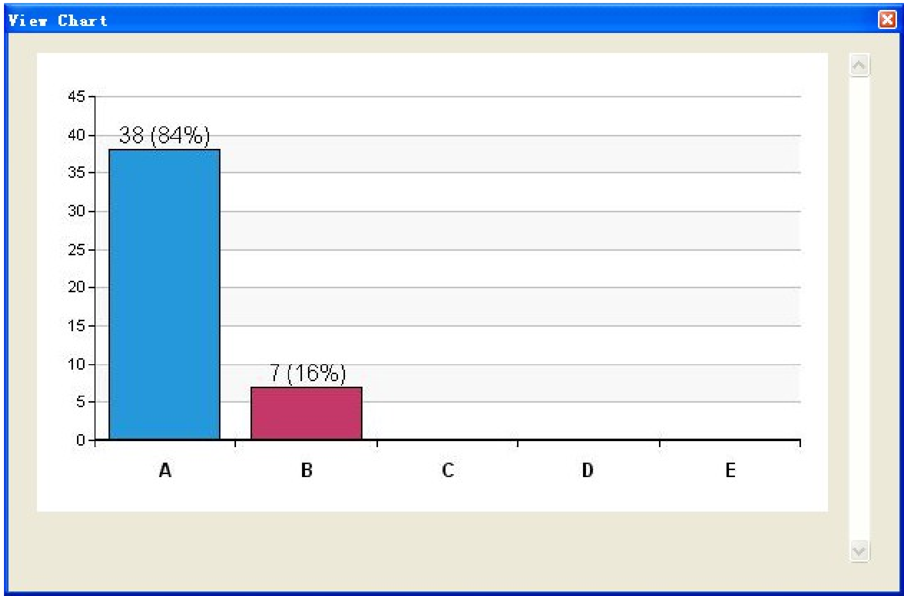
- 哪个 DFA 识别 {ε} ？



A

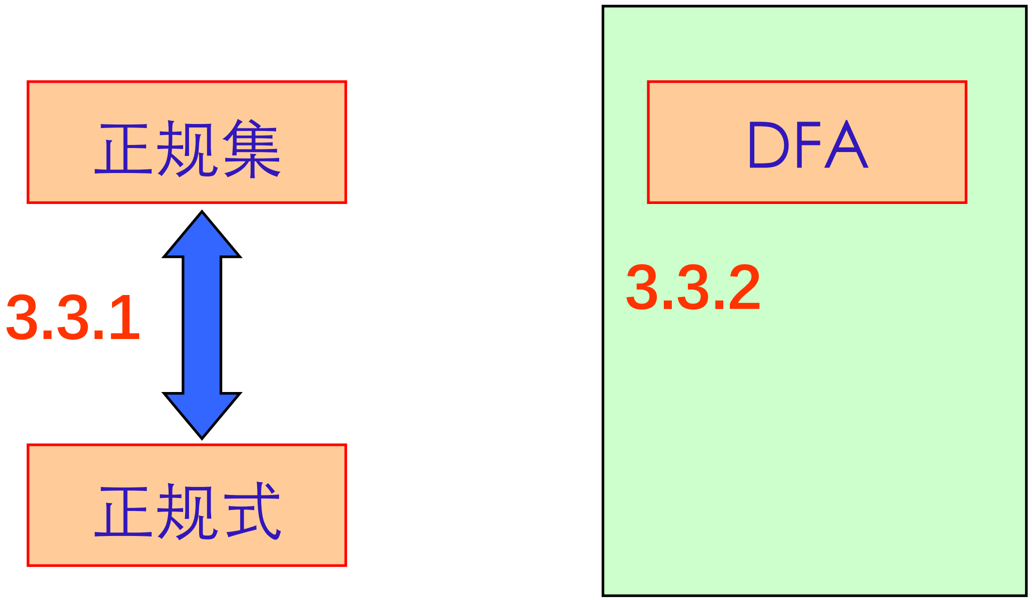


B



iclicker.

关系图



- 将证明：Σ上的字集 $V \subseteq \Sigma^*$ 是正规集，当且仅当存在Σ上的 DFA M，使得 $V = L(M)$

编译原理

第三章 词法分析

第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

回顾

DIM,IF, DO,STOP,END
number, name, age
125, 2169
…

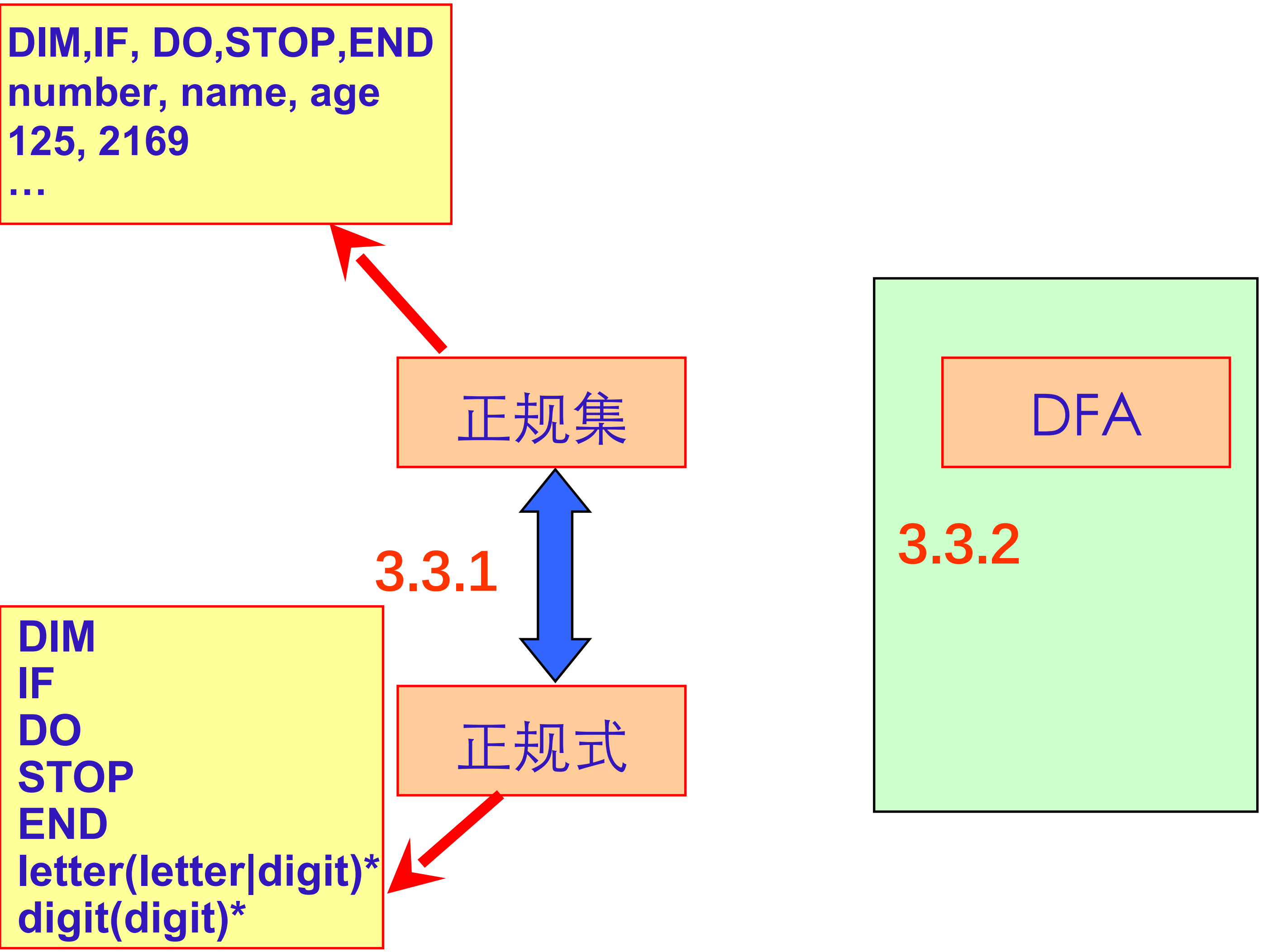
正规集

3.3.1

正规式

DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

关系图



DIM,IF, DO,STOP,END
number, name, age
125, 2169
…
正规集
3.3.1
正规式
DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*
DFA
3.3.2

确定有限自动机 (DFA)

- 对状态图进行形式化，则可以下定义：

确定有限自动机 (DFA)M 是一个五元式 $M=(S, \Sigma, f, S_0, F)$，其中：

1. S：有穷状态集
2. Σ：输入字母表（有穷）
3. f：状态转换函数，为 $S\times\Sigma\rightarrow S$ 的单值部分映射，$f(s, a)=s'$ 表示：当现行状态为 s，输入字符为 a 时，将状态转换到下一状态 s'，s' 称为 s 的一个后继状态
4. $S_0\in S$ 是唯一的一个初态
5. $F\subseteq S$ ：终态集（可空）

- 例如： DFA M=({0，1，2，3}，{a，b}，f，0，{3})，其中：f定义如下：

  f(0，a)=1　　f(0，b)=2

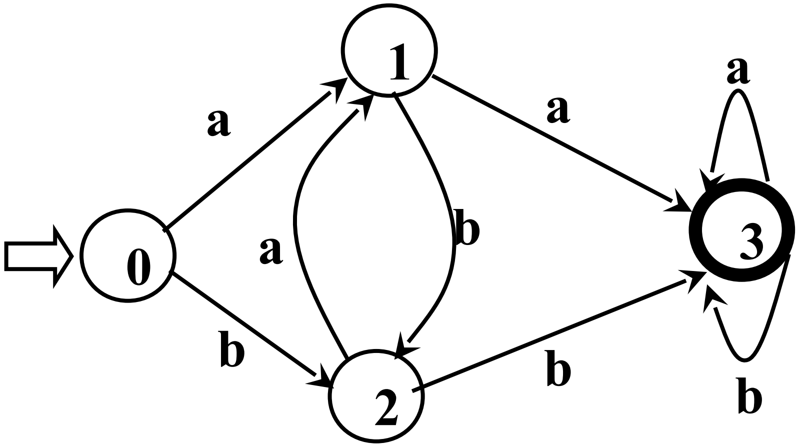
  f(1，a)=3　　f(1，b)=2

  f(2，a)=1　　f(2，b)=3

  f(3，a)=3　　f(3，b)=3

	a	b
0	1	2
1	3	2
2	1	3
3	3	3

状态转换矩阵



状态转换图

关系图

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then  返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169,
…

正规集

3.3.1

正规式

DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

FA

DFA

3.3.2
3.3.3

NFA

3.3.3 非确定有限自动机 (NFA)

- 1976 年图灵奖
  - For their joint paper "Finite Automata and Their Decision Problem," which introduced the idea of nondeterministic machines, which has proved to be an enormously valuable concept. Their (Scott & Rabin) classic paper has been a continuous source of inspiration for subsequent work in this field.


M. O. Rabin*
D. Scott†

Finite Automata and Their Decision Problems‡

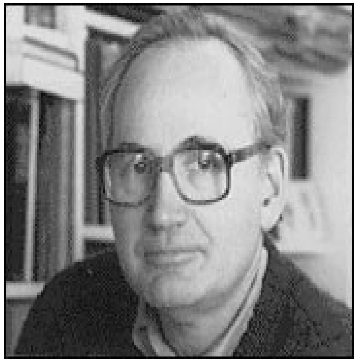
Abstract: Finite automata are considered in this paper as instruments for classifying finite tapes. Each one-tape automaton defines a set of tapes, a two-tape automaton defines a set of pairs of tapes, et cetera. The structure of the defined sets is studied. Various generalizations of the notion of an automaton are introduced and their relation to the classical automata is determined. Some decision problems concerning automata are shown to be solvable by effective algorithms; others turn out to be unsolvable by algorithms.

Introduction

114

IBM JOURNAL • APRIL 1959


Michael O. Rabin


Dana S. Scott

3.3.3 非确定有限自动机 (NFA)

- 定义：一个非确定有限自动机 (NFA) M 是一个五元式 M=(S, Σ, f, S_0, F)，其中：

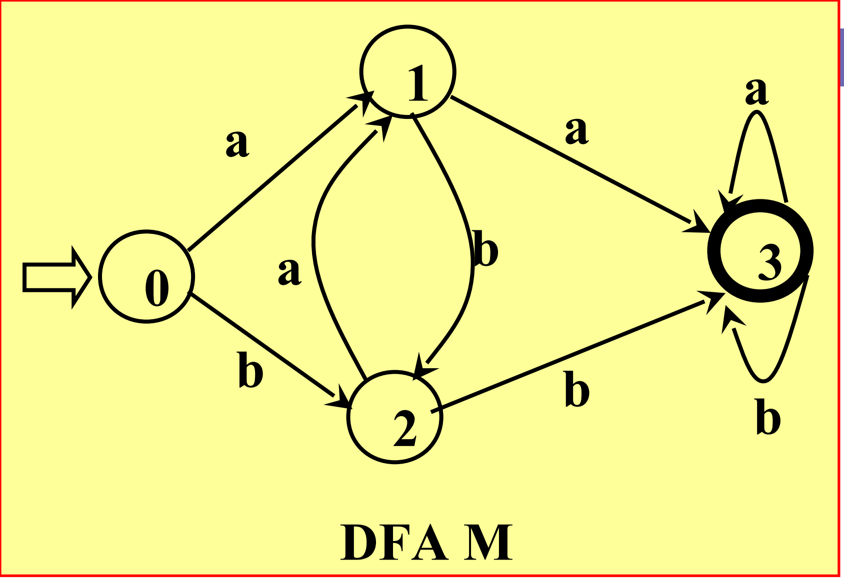
1 S: 有穷状态集

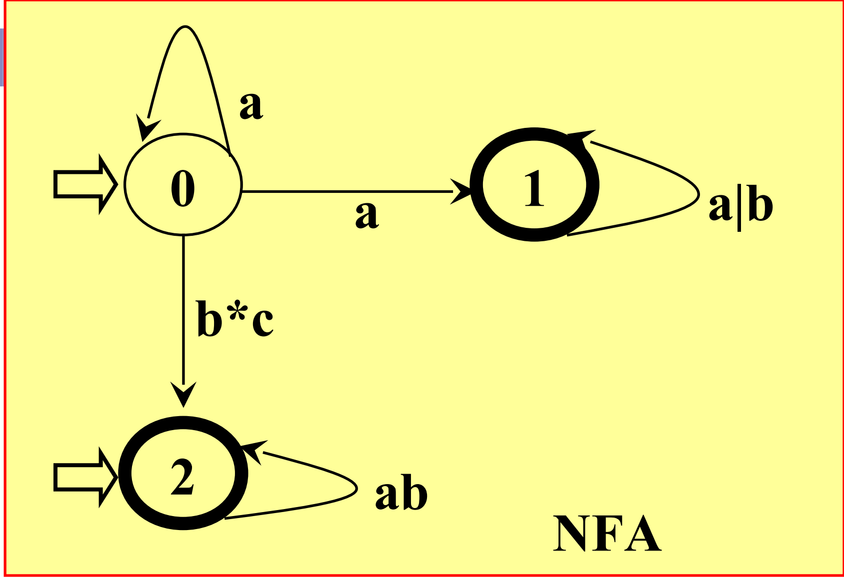
2 Σ ： 输入字母表 (有穷)

3 f: 状态转换函数，为 $S\times\Sigma^{*}\rightarrow 2^{S}$ 的部分映射

4 $S_0 \subseteq S$ 是非空的初态集

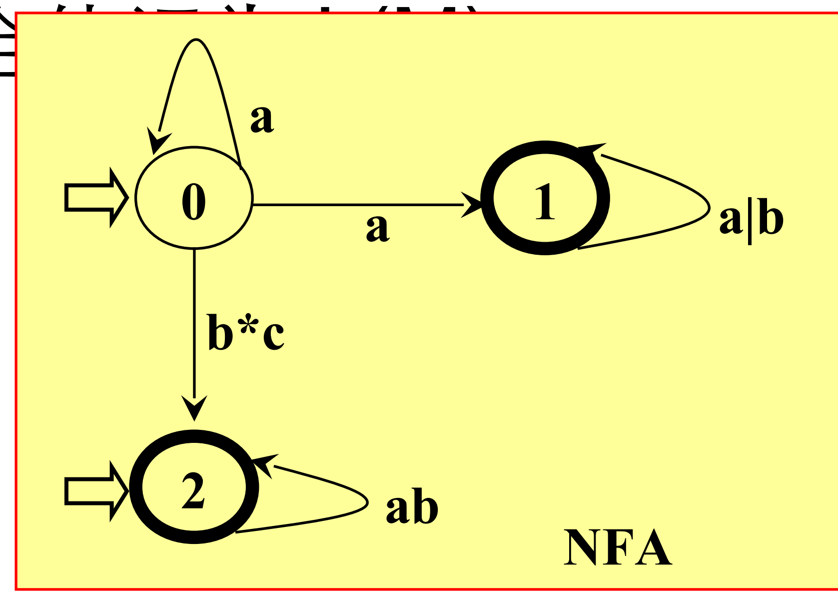
5 F $\subseteq S$ ： 终态集 (可空)



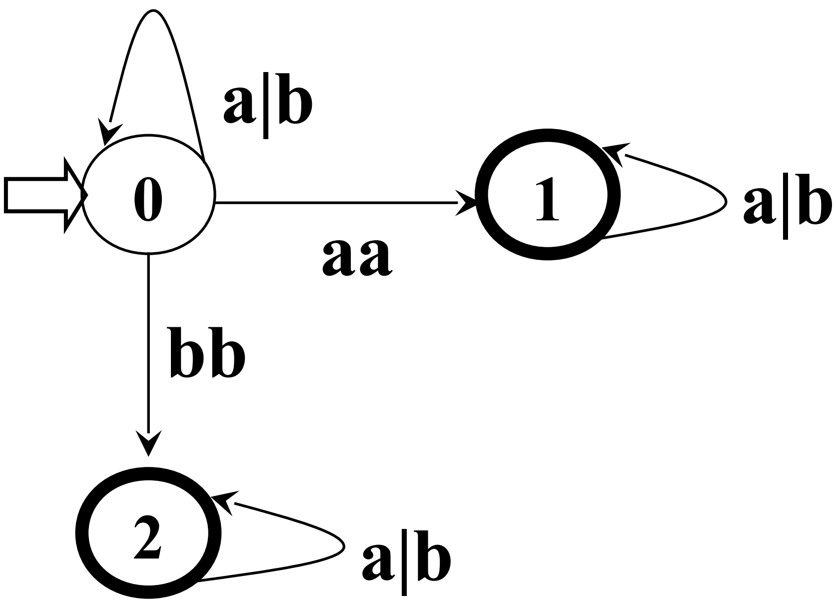


- 从状态图看 NFA 和 DFA 的区别
  - 可以有多个初态
  - 弧上的标记可以是Σ*中的一个字（甚至可以是一个正规式），而不一定是单个字符
  - 同一个字可能出现在同状态射出的多条弧上
- DFA 是 NFA 的特例

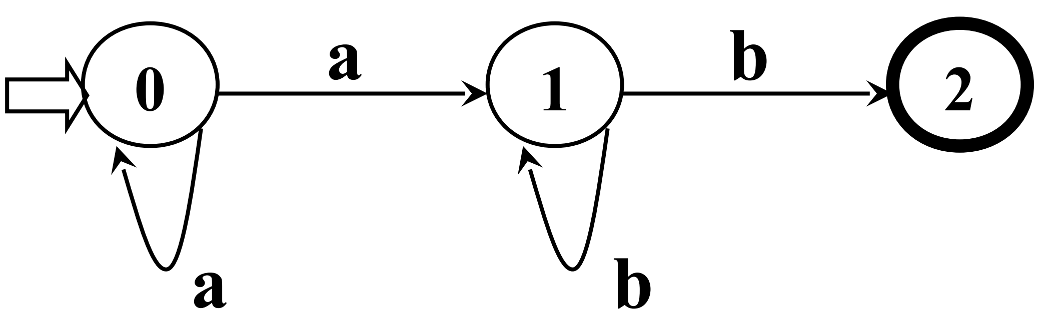
- 对于Σ * 中的任何字α，若存在一条从初态到某一终态的道路，且这条路上所有弧上的标记字连接成的字等于α（忽略那些标记为ε的弧），则称α为 NFA M 所识别（接收）
- NFA M 所识别的字的全



NFA 示例



识别所有含相继两个 a 或相继两个 b 的字



$\{a^m b^n \mid m, \quad n \geq 1\}$

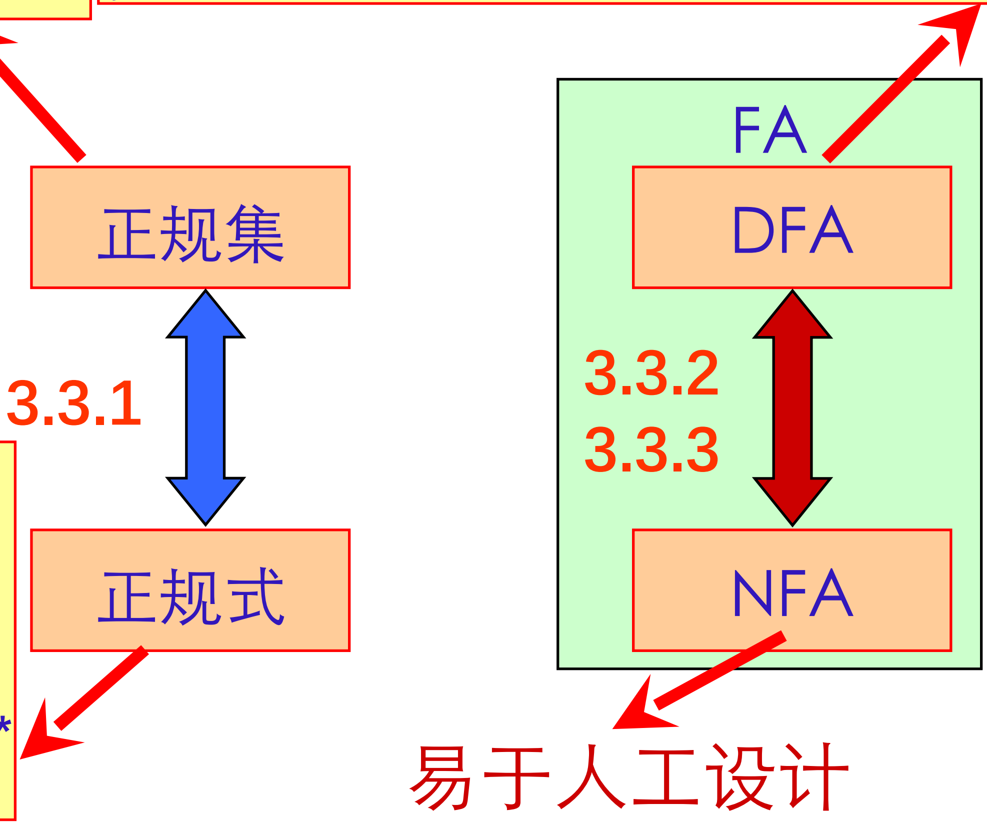
NFA 与 DFA

- 定义：对于任何两个有限自动机 M 和 M’，如果 L(M)=L(M’)，则称 M 与 M’ 等价
- 自动机理论中一个重要的结论：判定两个自动机等价性的算法是存在的
- 对于每个 NFA M 存在一个 DFA M’，使得 L(M)=L(M’)
- DFA 与 NFA 描述能力相同！

关系图

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then  返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169
…



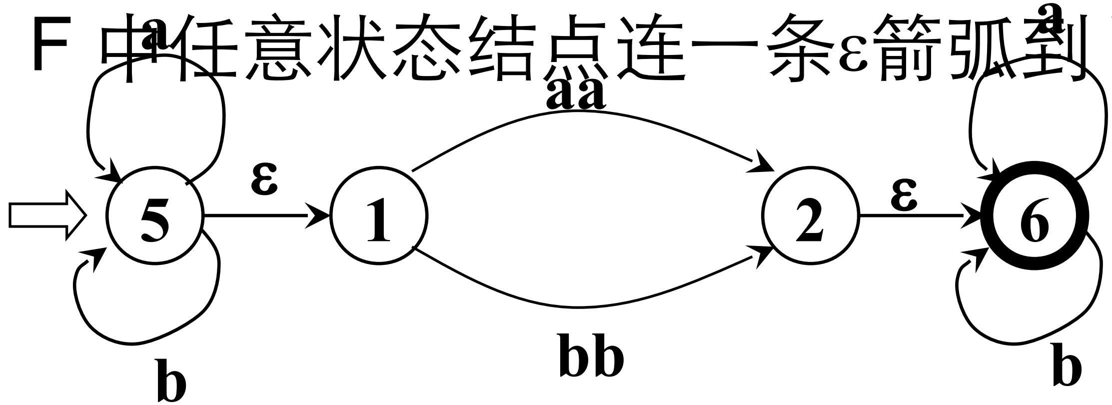
DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

证明：

1. 假定 NFA M=<S, Σ, δ, S_0, F>，我们对 M 的状态转换图进行以下改造:

   1) 引进新的初态结点 X 和终态结点 Y， X,Y∉S，

      从 X 到 S_0 中任意状态结点连一条ε箭弧， 从 F 中任意状态结点连一条ε箭弧到 Y 。

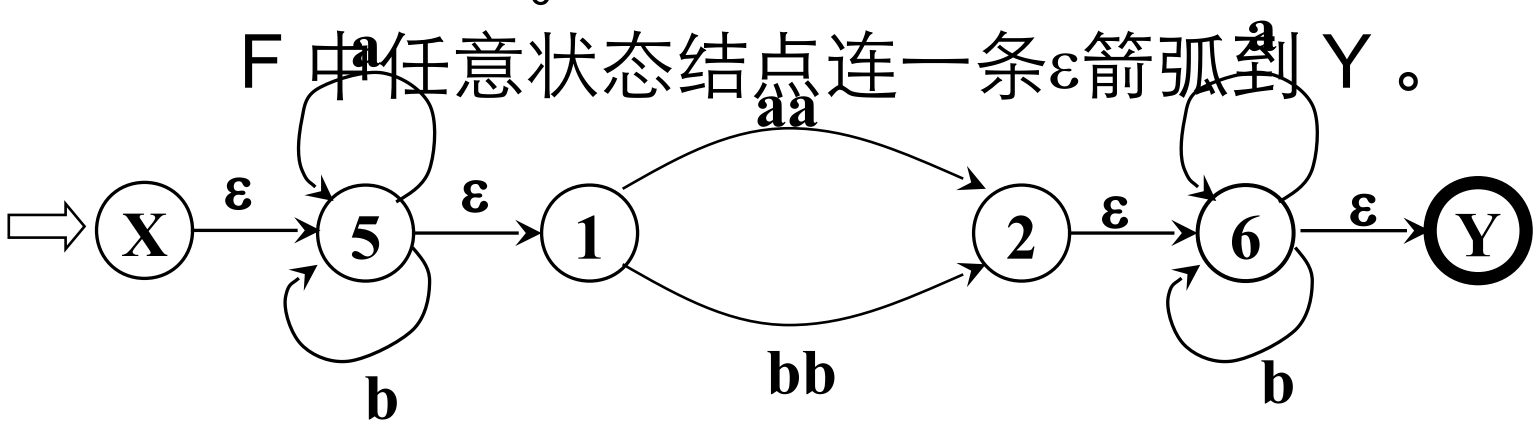


证明：

1. 假定 NFA M=<S, Σ, δ, S_0, F>， 我们对 M 的状态转换图进行以下改造:

   1) 引进新的初态结点 X 和终态结点 Y， X,Y∉S，

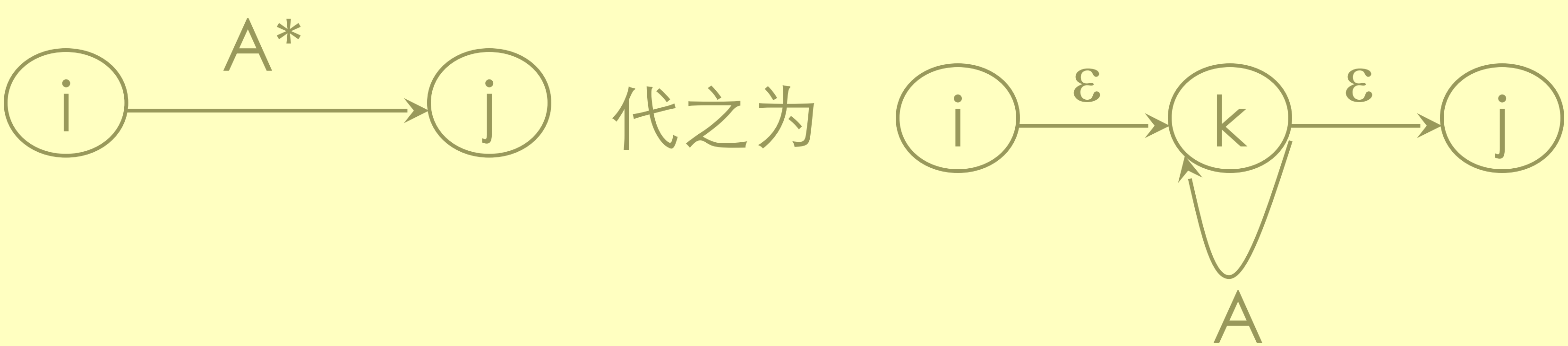
   从 X 到 S_0 中任意状态结点连一条ε箭弧， 从 F 中任意状态结点连一条ε箭弧到 Y 。



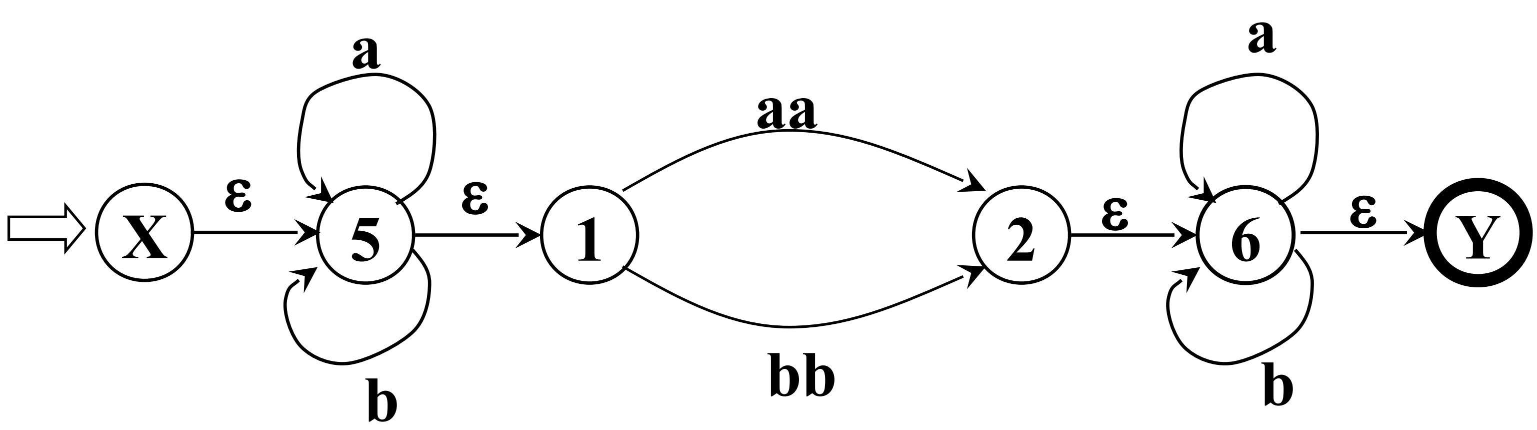
证明：

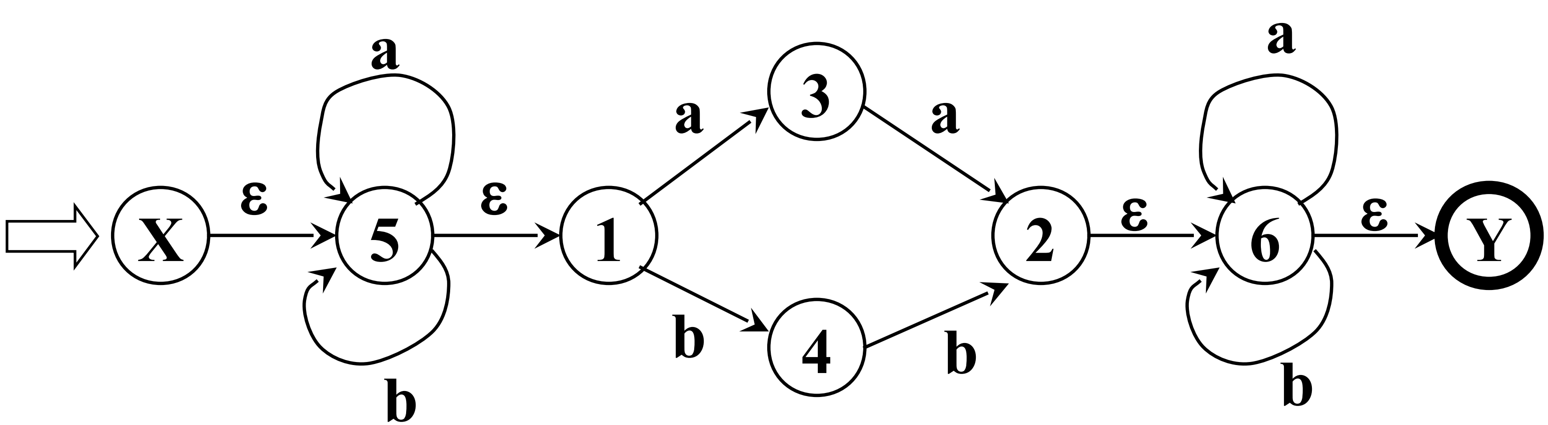
1. 假定 NFA M=<S, Σ, δ, S_0, F>，我们对 M 的状态转换图进行以下改造:
   1) 引进新的初态结点 X 和终态结点 Y， $X,Y \notin S$
      ，
      从 X 到 S_0 中任意状态结点连一条ε箭弧，从 F 中任意状态结点连一条ε箭弧到 Y。
   2) 对 M 的状态转换图进一步施行替换，其中 k 是新引入的状态。

按下面的三条规则对箭弧进行分裂：

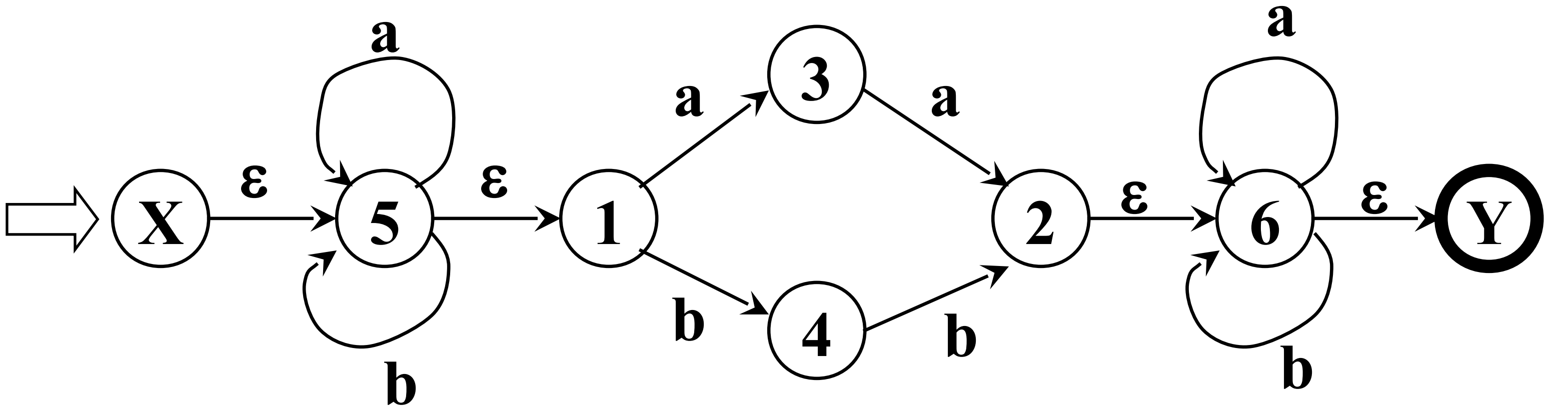


- 识别所有含相继两个 a 或相继两个 b 的字





- 逐步把这个图转变为每条弧只标记为Σ上的一个字符或ε，最后得到一个 NFA M’，显然 L(M’)=L(M)



2. 把上述 NFA 确定化——采用子集法

设 I 是的状态集的一个子集，定义 I 的ε - 闭包ε - closure(I) 为：

i) 若 s∈I，则 s∈ε-closure(I)；

ii) 若 s∈I，则从 s 出发经过任意条ε弧而能到达的任何状态 s’ 都属于ε -closure(I)

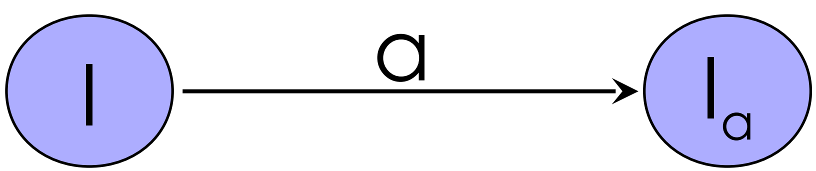
即

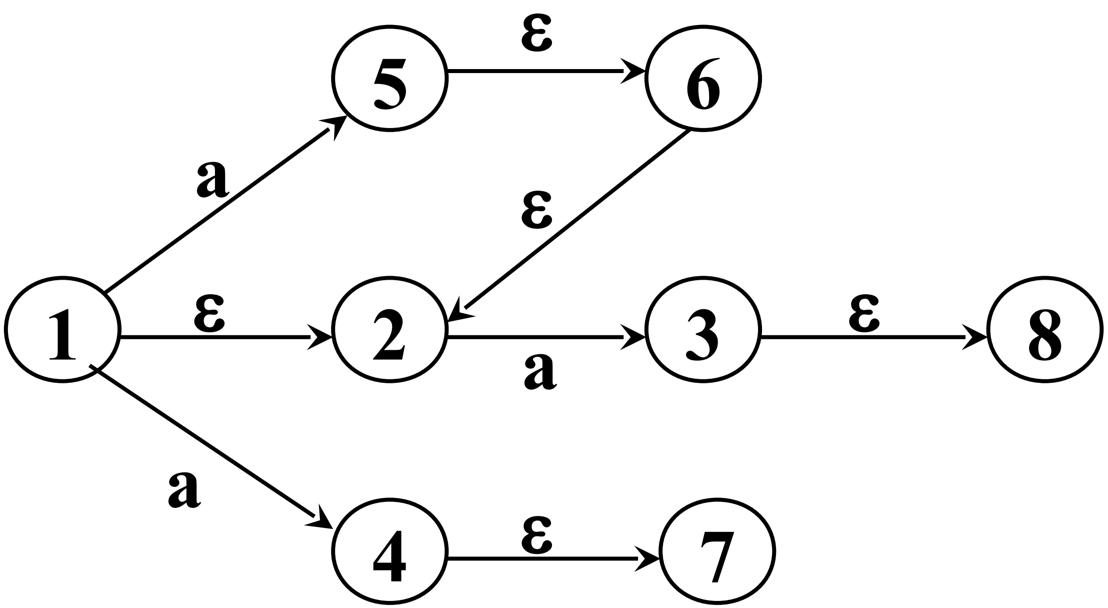
ε-closure(I)=I∪{s’ | 从某个 s∈I 出发经过任意条ε弧能到达 s’}

- 设 a 是Σ中的一个字符，定义

$$I_a = \varepsilon\text{-closure}(J)$$

其中，J 为 I 中的某个状态出发经过一条 a 弧而到达的状态集合。





- 设 a 是Σ中的一个字符，定义

  $I_a = \varepsilon\text{-closure}(J)$

  其中，J 为 I 中的某个状态出发经过一条 a 弧而到达的状态集合。

- ε-closure({1}) = {1， 2}= I

I_a?

J={5， 4， 3}

I_a= ε-closure(J) = ε-closure({5， 4， 3})

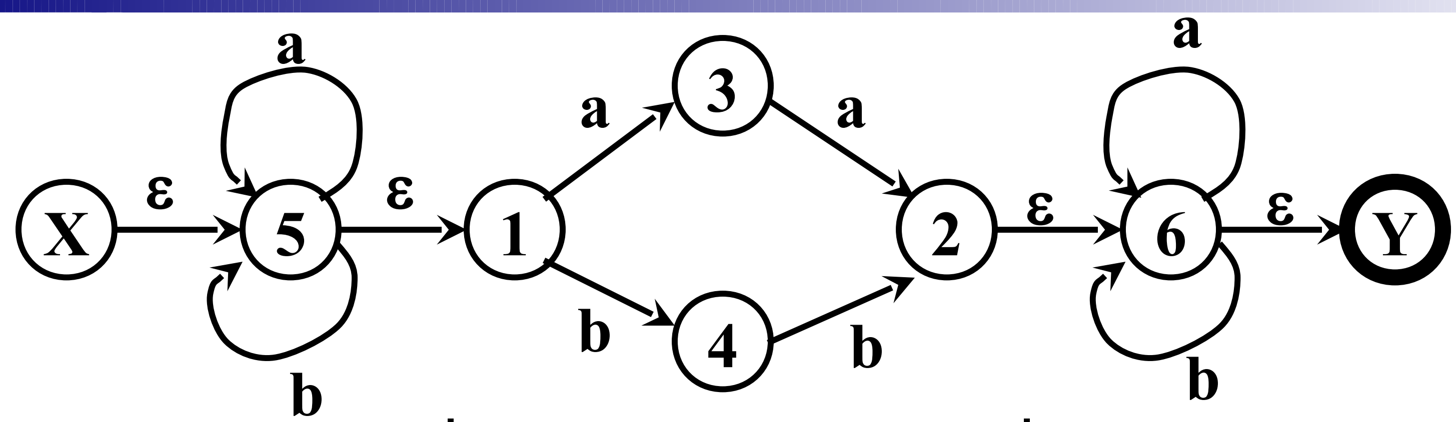
={5， 4， 3， 6， 2， 7， 8}

确定化的过程

- 不失一般性，设字母表只包含两个 *a* 和 *b*，我们构造一张表：

I	I_a	I_b
ε-Closure({X})	{…}	{…}
{…}	{…}	{…}
{…}	{…}	{…}

- 首先，置第 1 行第 1 列为 ε-closure({X}) 求出这一列的 I_a，I_b；
- 然后，检查这两个 I_a，I_b，看它们是否已在表中的第一列中出现，把未曾出现的填入后面的空行的第 1 列上，求出每行第 2，3 列上的集合 ...
- 重复上述过程，直到所有第 2，3 列子集全部出现在第一列为止



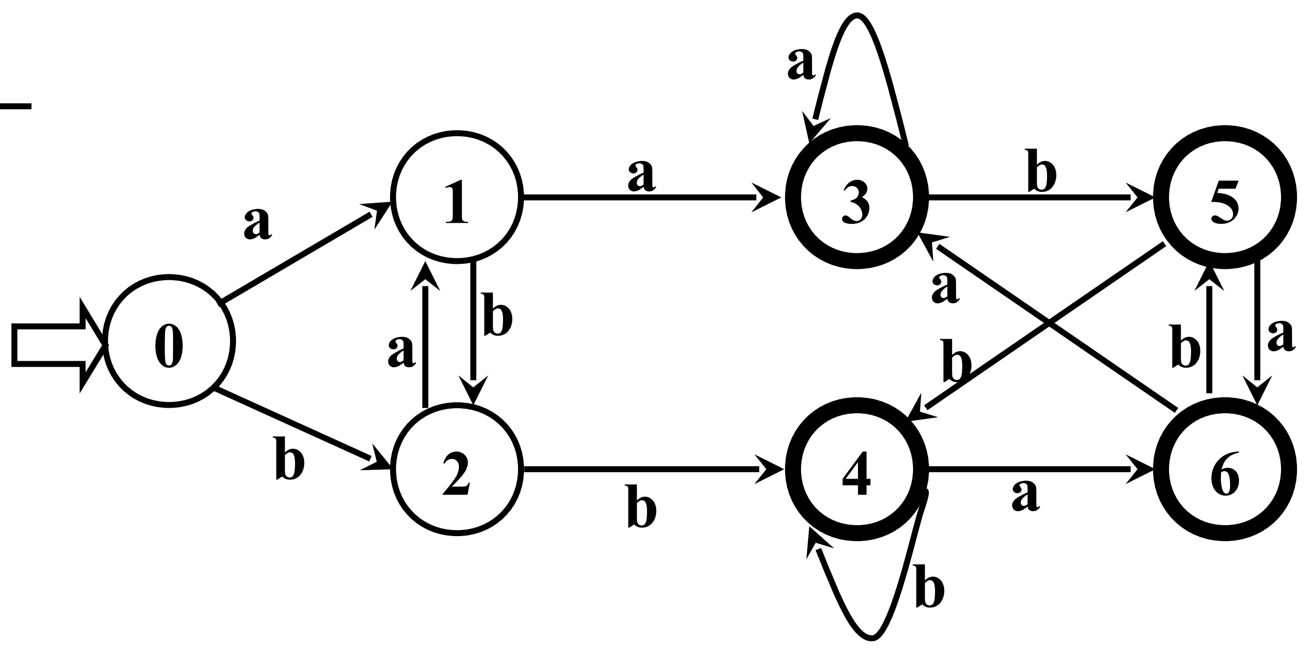
I	I_a	I_b
{X,5,1}	{5,3,1}	{5,4,1}
{5,3,1}	{5,2,3,1,6,Y}	{5,4,1}
{5,4,1}	{5,3,1}	{5,2,4,1,6,Y}
{5,2,3,1,6,Y}	{5,2,3,1,6,Y}	{5,4,6,1,Y}
{5,2,4,1,6,Y}	{5,3,6,1,Y}	{5,2,4,1,6,Y}
{5,4,6,1,Y}	{5,3,6,1,Y}	{5,2,4,1,6,Y}
{5,3,6,1,Y}	{5,2,3,1,6,Y}	{5,4,6,1,Y}

I	I_a	I_b
{X,5,1}	{5,3,1}	{5,4,1}
{5,3,1}	{5,2,3,1,6,Y}	{5,4,1}
{5,4,1}	{5,3,1}	{5,2,4,1,6,Y}
{5,2,3,1,6,Y}	{5,2,3,1,6,Y}	{5,4,6,1,Y}
{5,2,4,1,6,Y}	{5,3,6,1,Y}	{5,2,4,1,6,Y}
{5,4,6,1,Y}	{5,3,6,1,Y}	{5,2,4,1,6,Y}
{5,3,6,1,Y}	{5,2,3,1,6,Y}	{5,4,6,1,Y}

- 把这张表看成一个状态转换矩阵，把其中的每个子集看成一个状态
- 这张表唯一刻划了一个确定的有限自动机 M
  - 初态是ε -closure({X})
  - 终态是含有原终态 Y 的子集
- 不难看出，这个 DFA M 与 M’ 等价

I	I_a	I_b
{X,5,1}	{5,3,1}	{5,4,1}
{5,3,1}	{5,2,3,1,6,Y}	{5,4,1}
{5,4,1}	{5,3,1}	{5,2,4,1,6,Y}
{5,2,3,1,6,Y}	{5,2,3,1,6,Y}	{5,4,6,1,Y}
{5,2,4,1,6,Y}	{5,3,6,1,Y}	{5,2,4,1,6,Y}
{5,4,6,1,Y}	{5,3,6,1,Y}	{5,2,4,1,6,Y}
{5,3,6,1,Y}	{5,2,3,1,6,Y}	{5,4,6,1,Y}

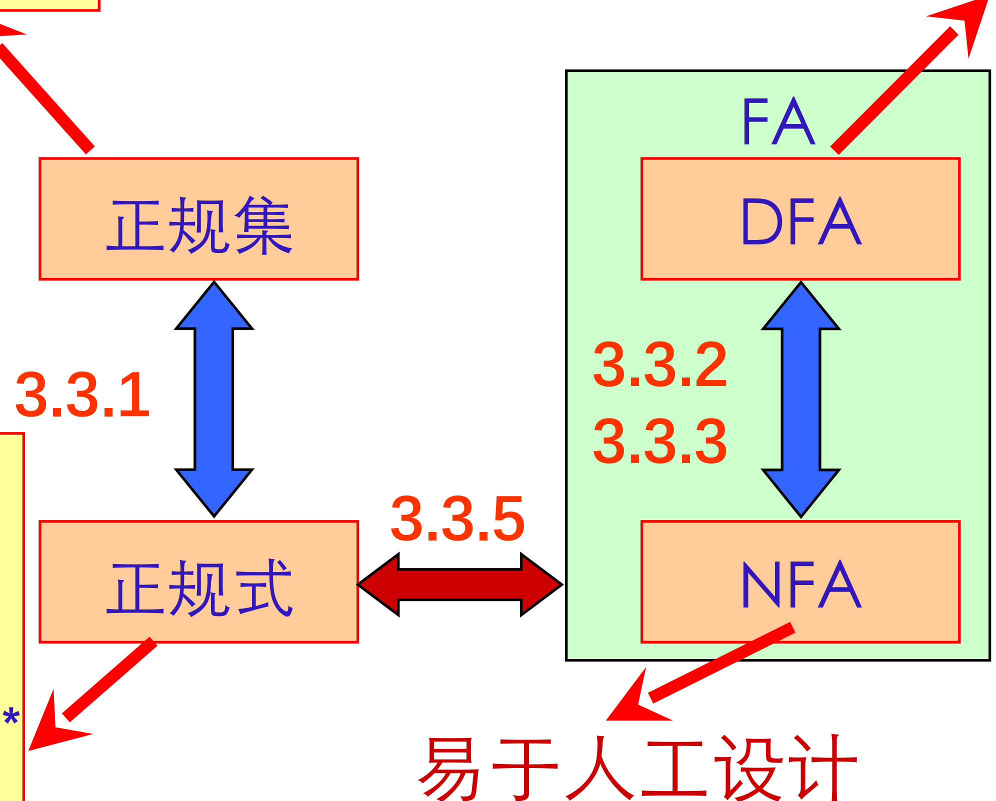
I	a	b
0	1	2
1	3	2
2	1	4
3	3	5
4	6	4
5	6	4
6	3	5



小结

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then  返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169
…



DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

编译原理

第三章 词法分析

第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

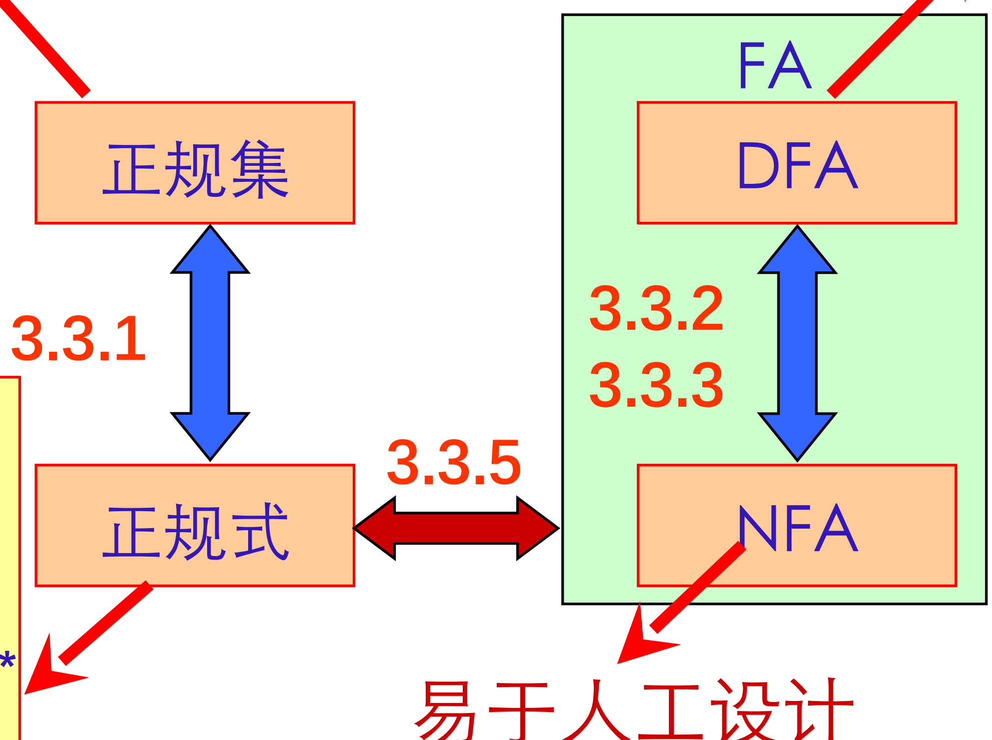
第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

关系图

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then  返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169,
…



DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

3.3.5 正规式与有限自动机的等价性

- 定理:

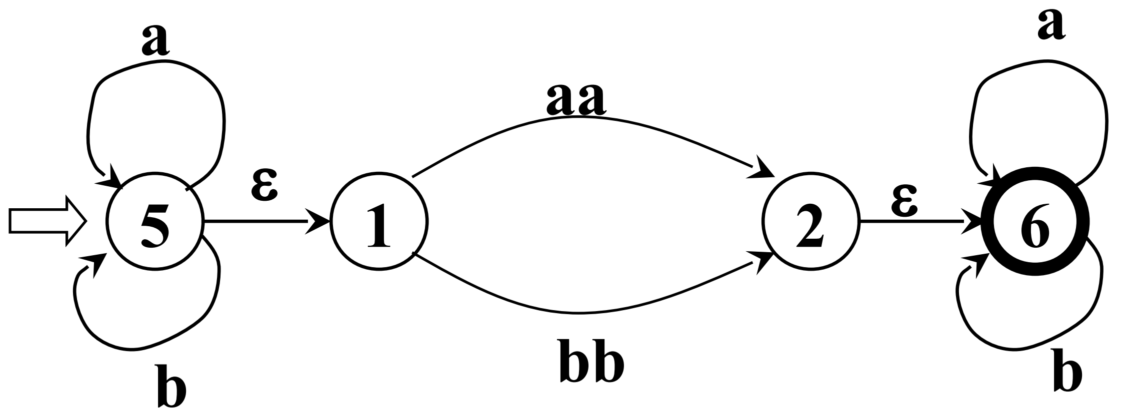
1. 对任何 FA M，都存在一个正规式 r，使得 L(r)=L(M)。
2. 对任何正规式 r，都存在一个 FA M，使得 L(M)=L(r)。

- 对转换图概念拓广，令每条弧可用一个正规式作标记。（对一类输入符号）

- 证明：

1 对Σ上任一 NFA M ，构造一个Σ上的正规式 r ，使得 L(r)=L(M) 。

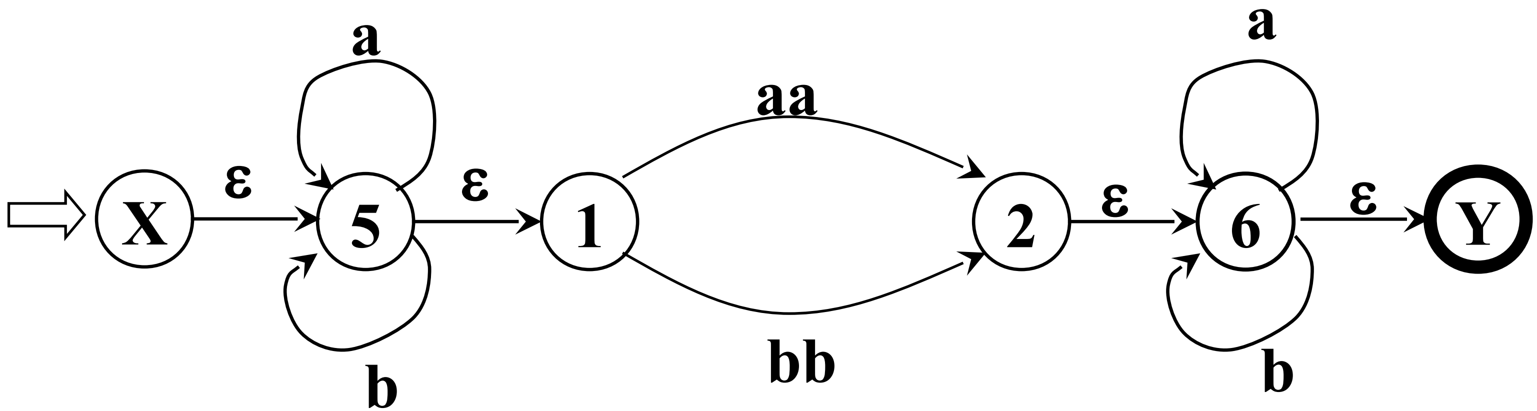
  - 首先，在 M 的转换图上加进两个状态 X 和 Y ，从 X 用ε弧连接到 M 的所有初态结点，从 M 的所有终态结点用ε弧连接到 Y ，从而形成一个新的 NFA ，记为 M' ，它只有一个初态 X 和一个终态 Y ，显然 L(M)=L(M') 。



- 证明:

1 对Σ上任一 NFA M，构造一个Σ上的正规式 r，使得 L(r)=L(M)。

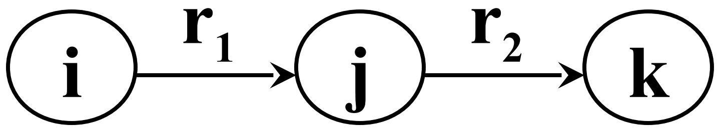
  - 首先，在 M 的转换图上加进两个状态 X 和 Y，从 X 用ε弧连接到 M 的所有初态结点，从 M 的所有终态结点用ε弧连接到 Y，从而形成一个新的 NFA，记为 M’，它只有一个初态 X 和一个终态 Y，显然 L(M)=L(M’)。



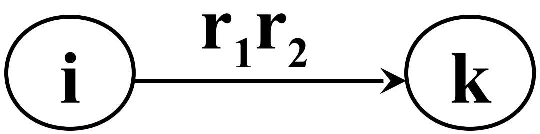
- 证明:

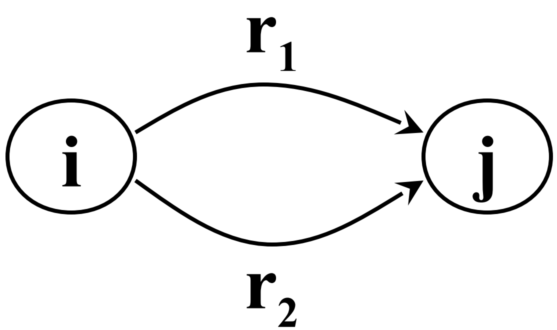
1 对Σ上任一 NFA M，构造一个Σ上的正规式 r，使得 L(r)=L(M)。

  - 首先，在 M 的转换图上加进两个状态 X 和 Y，从 X 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，从而形成一个新的 NFA，记为 M’，它只有一个初态 X 和一个终态 Y，显然 L(M)=L(M’)。
  - 然后，反复使用下面的一条规则，逐步消去的所有结点，直到只剩下 X 和 Y 为止;

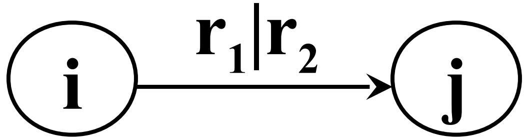


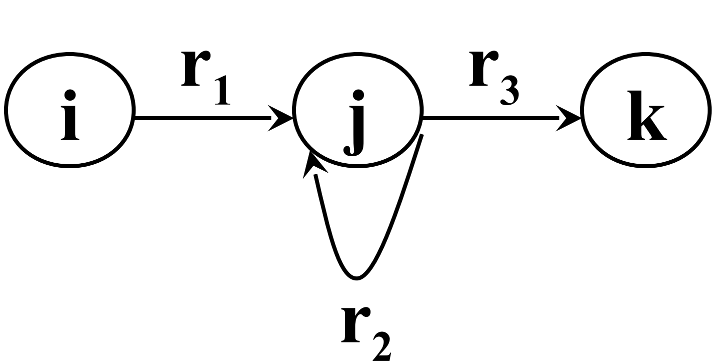
代之为



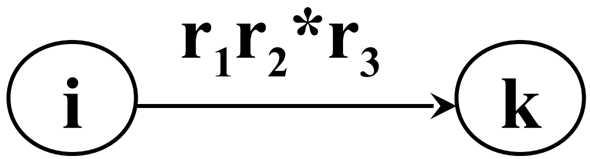


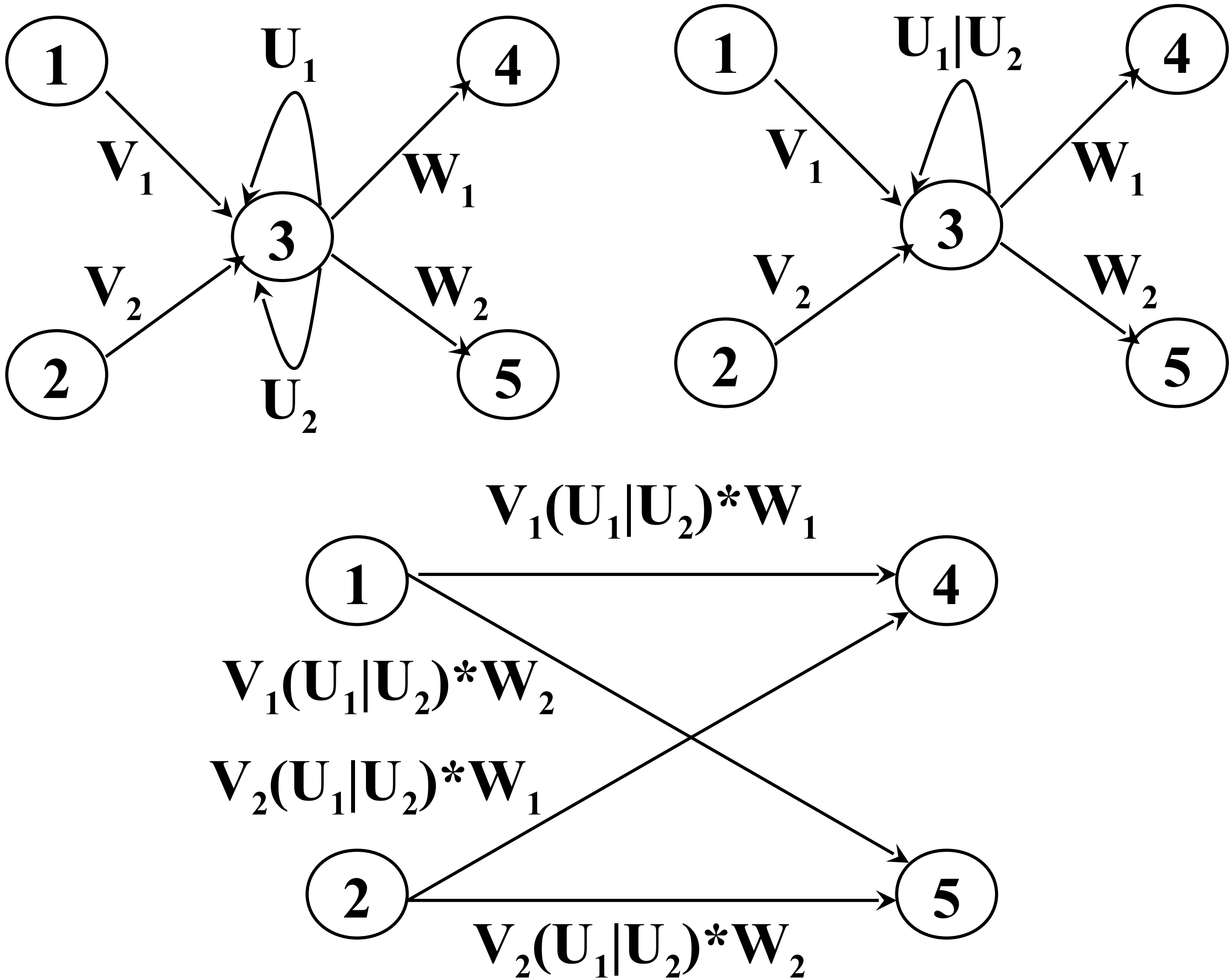
代之为





代之为




1
U_1
4
V_1
W_1
3
V_2
W_2
2
U_2
5
1
$U_1|U_2$
4
V_1
W_1
3
V_2
W_2
2
5
$V_1(U_1|U_2)^*W_1$
1
4
$V_1(U_1|U_2)^*W_2$
$V_2(U_1|U_2)^*W_1$
2
5
$V_2(U_1|U_2)^*W_2$

- 最后， X 到 Y 的弧上标记的正规式即为所构造的正规式 r
- 显然 L(r)=L(M)=L(M'）

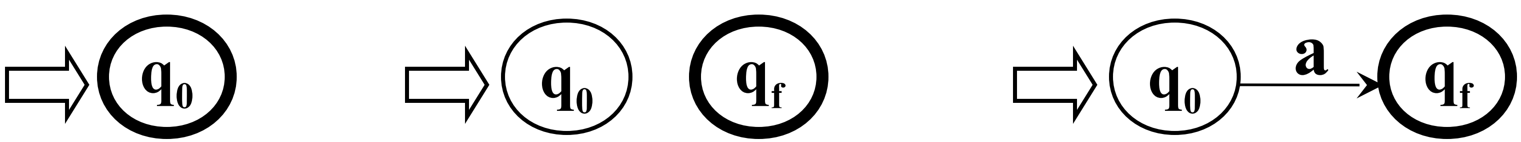
1．对任何 FA M，都存在一个正规式 r，使得 L(r)=L(M)。

2．对任何正规式 r，都存在一个 FA M，使得 L(M)=L(r)。

- 证明 2: 对于Σ上的正规式 r， 构造一个 NFA M， 使 L(M)=L(r)， 并且 M 只有一个终态，而且没有从该终态出发的箭弧。

下面使用关于 r 中运算符数目的归纳法证明上述结论。

(1) 若 r 具有零个运算符，则 r=ε 或 r=ϕ 或 r=a，其中 a∈Σ 。此时下图所示的三个有限自动机显然符合上述要求。



(2) 假设结论对于少于 k(k≥1) 个运算符的正规式成立。

当 r 中含有 k 个运算符时， r 有三种情形：

- 情形 1： $r=r_1|r_2$， r_1 和 r_2 中运算符个数少于 k 。从而，由归纳假设，对 r_i 存在 $M_i=<S_i, \Sigma_i, \delta_i, q_i, \{f_i\}>$， 使得 $L(M_i)=L(r_i)$， 并且 M_i 没有从终态出发的箭弧（ i=1,2 ） 。不妨设 $S_1 \cap S_2=\phi$， 在 $S_1 \cup S_2$ 中加入两个新状态 q_0， f_0。

令 $M=<S_1 \cup S_2 \cup \{q_0,f_0\}, \Sigma_1 \cup \Sigma_2, \delta, q_0, \{f_0\}>$，其中δ定义如下：

(a) $\delta(q_0, \varepsilon)=\{q_1,q_2\}$

(b) $\delta(q,a)= \delta_1(q,a)$，当 $q\in S_1-\{f_1\}$, $a\in\Sigma_1 \cup \{\varepsilon\}$

(c) $\delta(q,a)= \delta_2(q,a)$，当 $q\in S_2-\{f_2\}$, $a\in\Sigma_2 \cup \{\varepsilon\}$

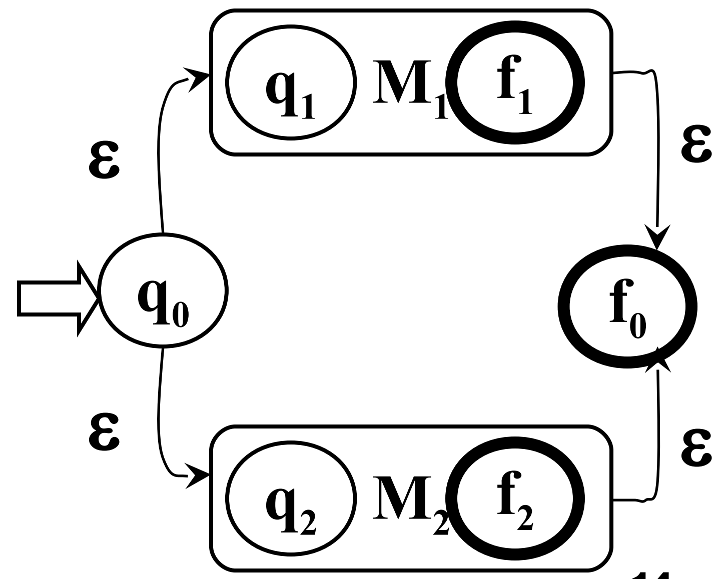
(d) $\delta(f_1,\varepsilon)=\delta(f_2,\varepsilon)=\{f_0\}$ 。

M 的状态转换如右图所示。

$L(M)=L(M_1) \cup L(M_2)$

$=L(r_1) \cup L(r_2)=L(r)$

$r=r_1|r_2$



- 情形 2： $r=r_1r_2$，设 M_i 同情形 1(i=1,2)。

  令 $M=<S_1\cup S_2, \Sigma_1\cup\Sigma_2, \delta, q_1, \{f_2\}>$，其中δ定义如下：

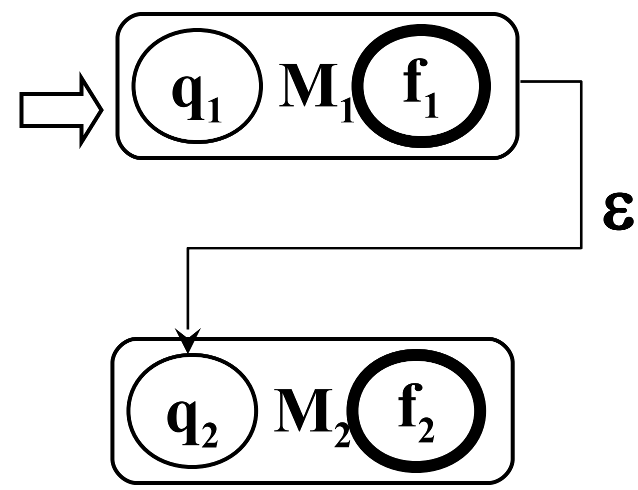
  (a) $\delta(q,a)= \delta_1(q,a)$，当 $q\in S_1-\{f_1\}, a\in\Sigma_1\cup\{\varepsilon\}$

  (b) $\delta(q,a)= \delta_2(q,a)$，当 $q\in S_2, a\in\Sigma_2\cup\{\varepsilon\}$

  (c) $\delta(f_1,\varepsilon)=\{q_2\}$

$r=r_1r_2$

M 的状态转换如右图所示。



$L(M)=L(M_1)L(M_2)$

$=L(r_1)L(r_2)=L(r)$。

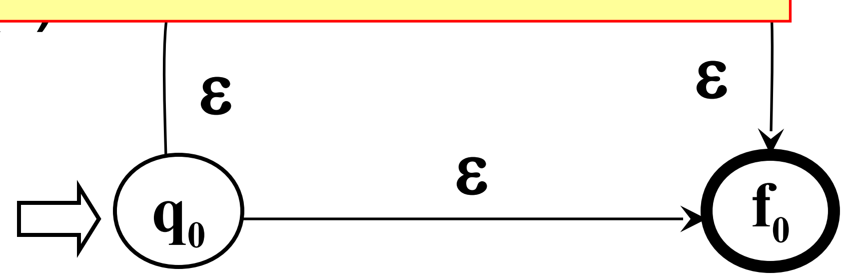
- 情形 3： $r=r_1^*$。设 M_1 同情形 1。

令 $M=<S_1 \cup \{q_0, f_0\}, \Sigma_1, \delta, q_0, \{f_0\}>$， 其中 $q_0, f_0 \notin S_1$， δ定义如下：

(a) $\delta(q_0,\varepsilon)=\delta(f_1,\varepsilon)=\{q_1, f_0\}$

1. 对任何 FA M， 都存在一个正规式 r， 使得 L(r)=L(M)。

2. 对任何正规式 r， 都存在一个 FA M， 使得 L(M)=L(r)。

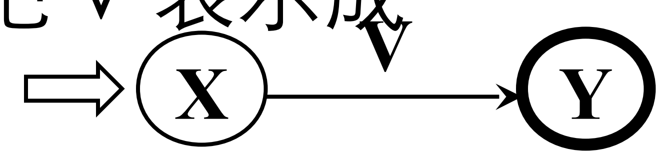


至此，结论 2 获证。

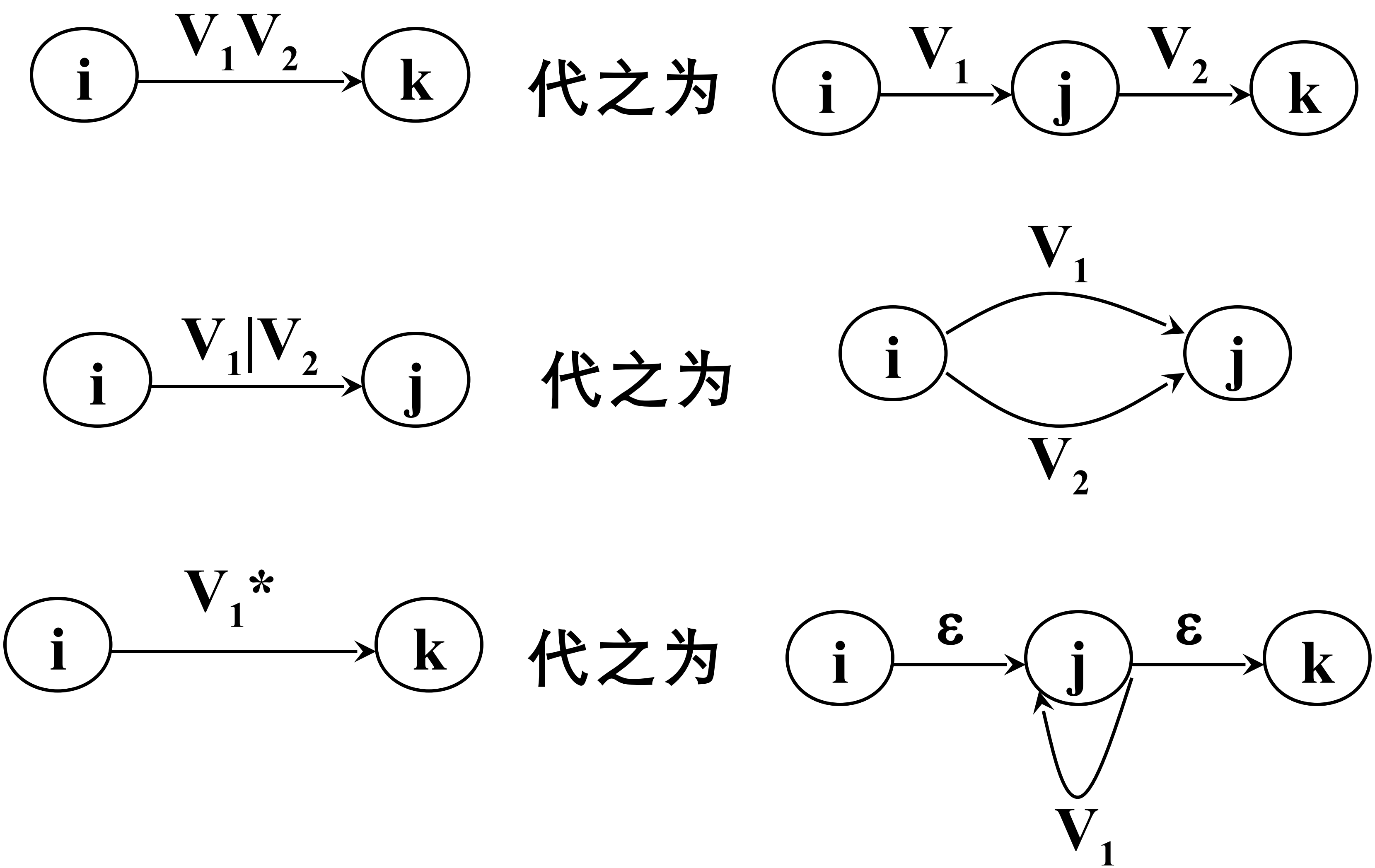
上述证明过程实质上是一个将正规表达式转换为有限自动机的算法

1）构造Σ上的 NFA M’ 使得 L(V)=L(M’)

首先，把 V 表示成

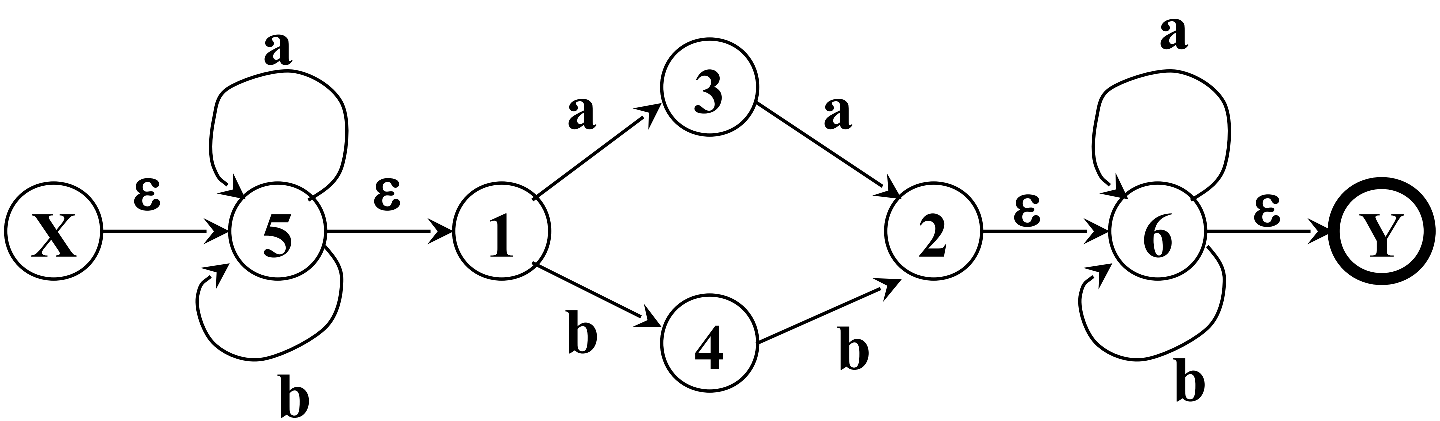


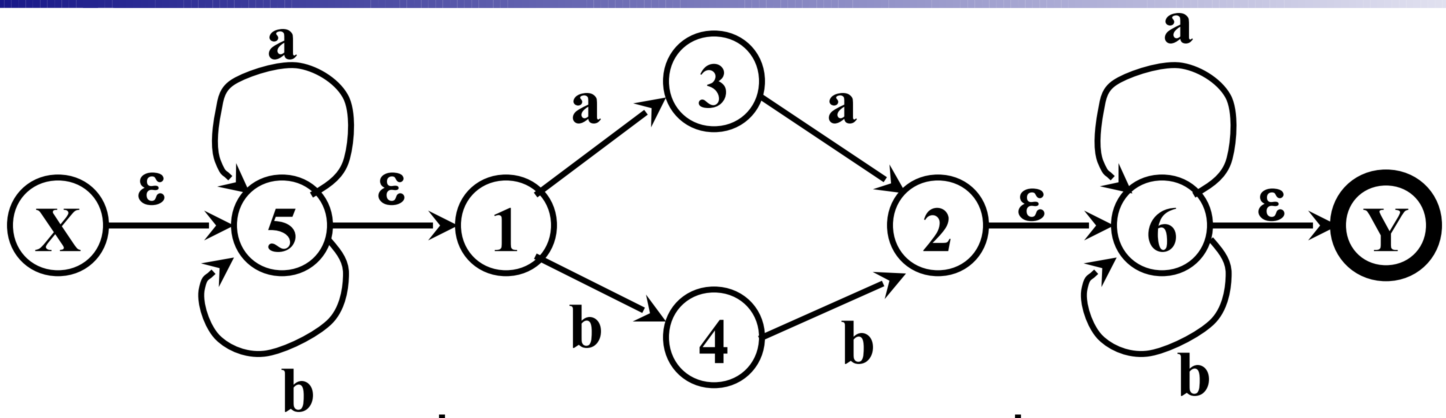
按下面的三条规则对 V 进行分裂



- 逐步把这个图转变为每条弧只标记为Σ上的一个字符或ε，最后得到一个 NFA M’，显然 L(M’)=L(V)

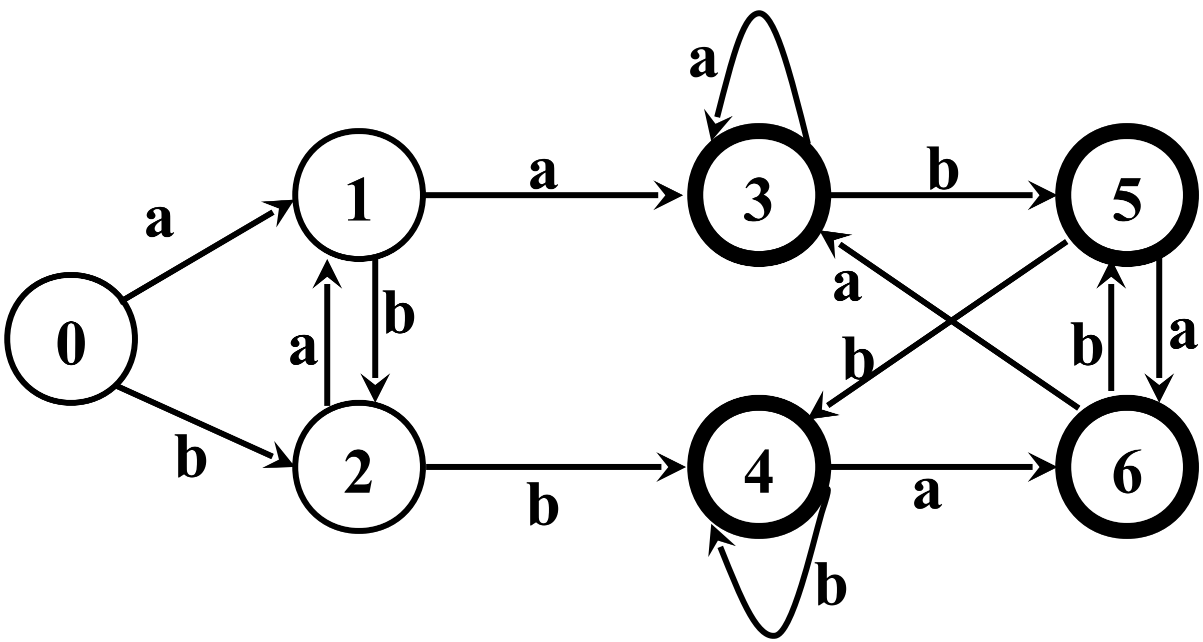
- (a|b)*(aa|bb)(a|b)*





I	I_a	I_b
{X,5,1}	{5,3,1}	{5,4,1}
{5,3,1}	{5,2,3,1,6,Y}	{5,4,1}
{5,4,1}	{5,3,1}	{5,2,4,1,6,Y}
{5,2,3,1,6,Y}	{5,2,3,1,6,Y}	{5,4,6,1,Y}
{5,2,4,1,6,Y}	{5,3,6,1,Y}	{5,2,4,1,6,Y}
{5,4,6,1,Y}	{5,3,6,1,Y}	{5,2,4,1,6,Y}
{5,3,6,1,Y}	{5,2,3,1,6,Y}	{5,4,6,1,Y}

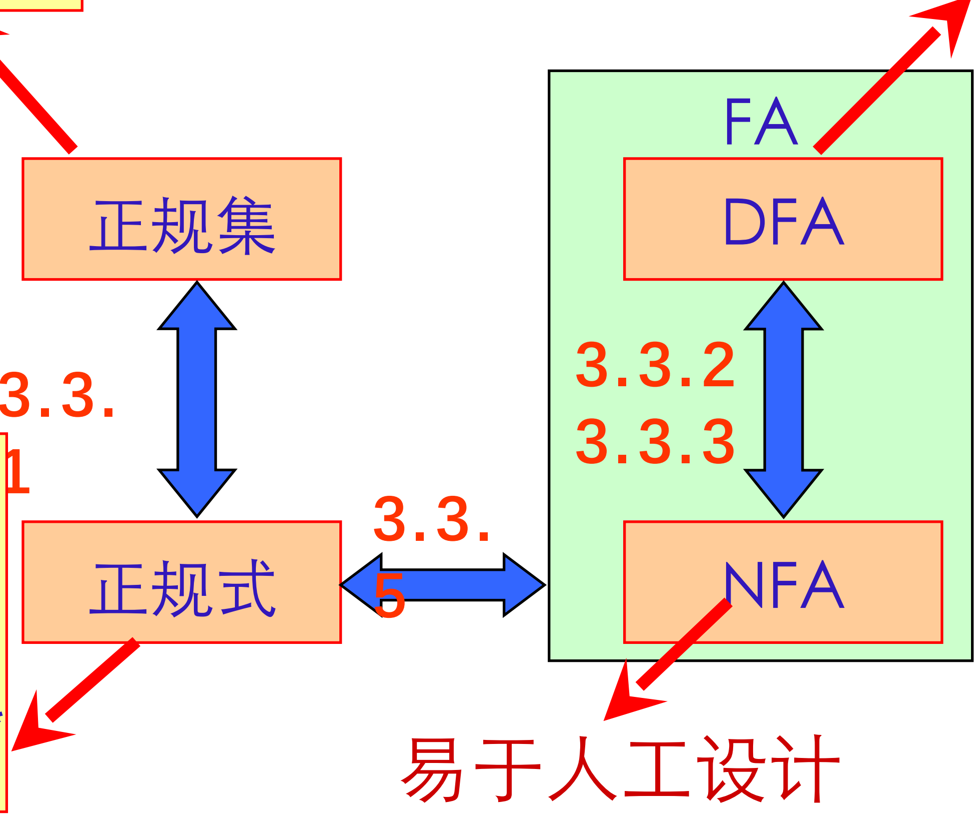
I	a	b
0	1	2
1	3	2
2	1	4
3	3	5
4	6	4
5	6	4
6	3	5



小结

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then  返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169
…



DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

编译原理

第三章 词法分析

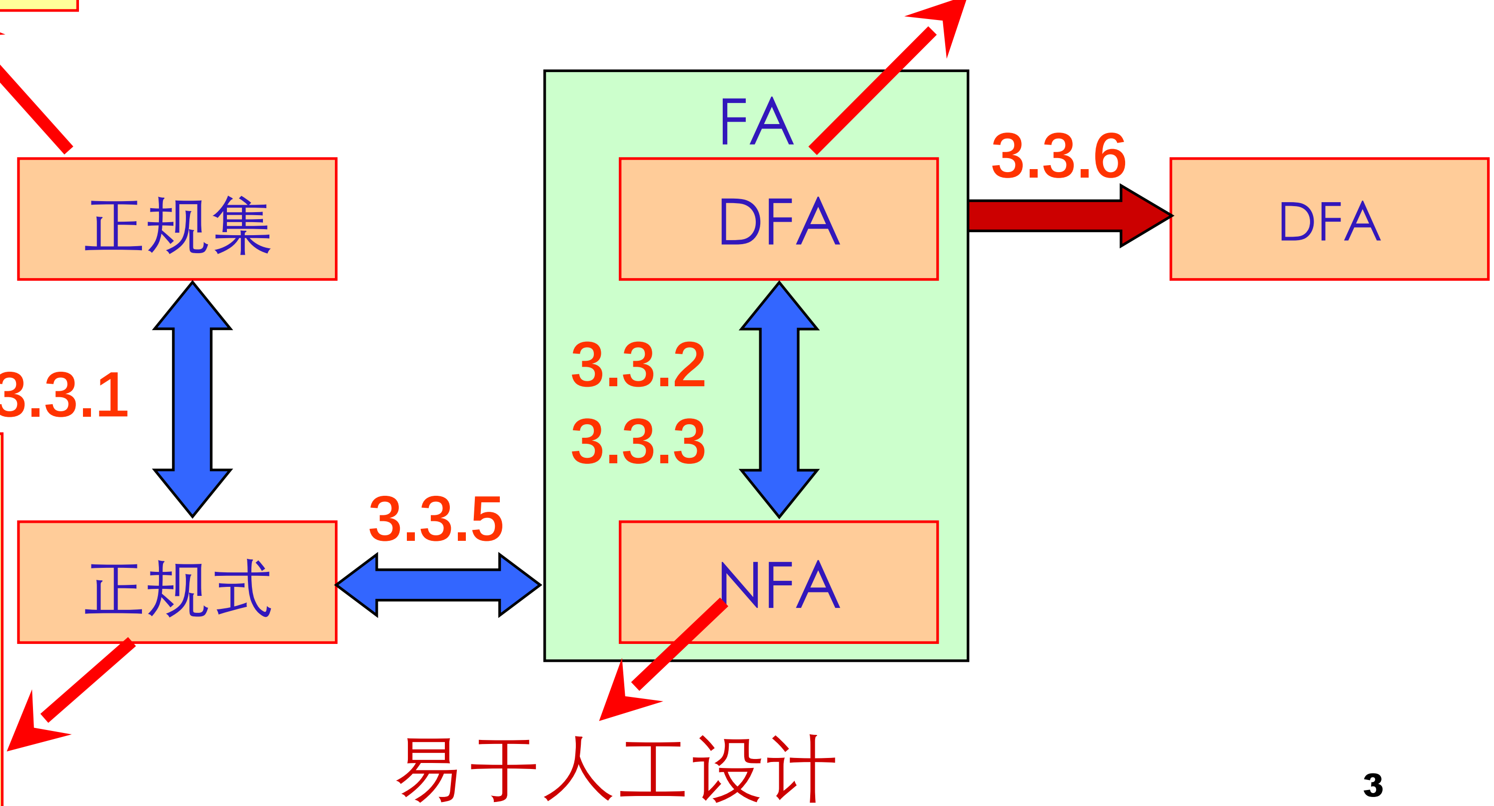
第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

关系图

DIM,IF, DO,STOP,END
number, name, age
125, 2169,
…

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then 返回 strToken 中的单
    GetChar( );
}
```



DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

3.3.6 确定有限自动机的化简

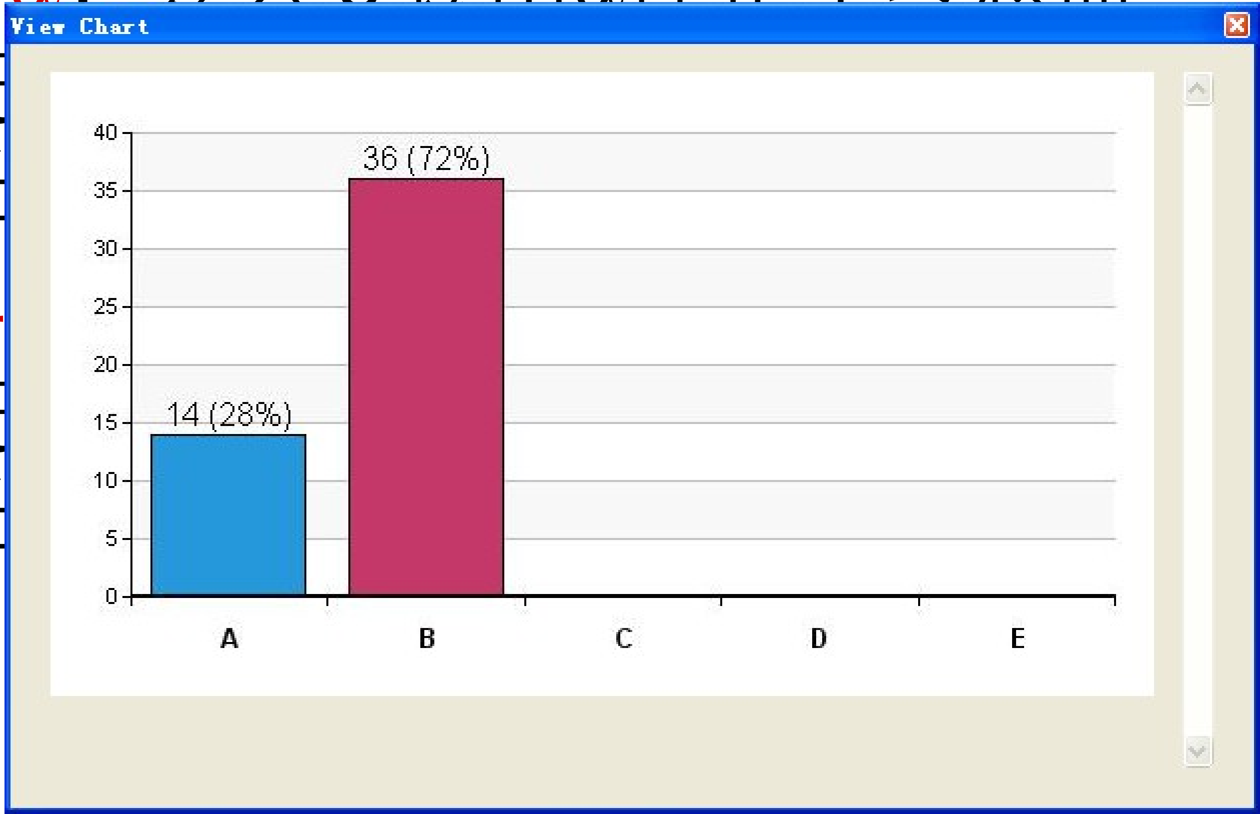
- 对 DFA M 的化简：寻找一个状态数比 M 少的 DFA M’，使得 L(M)=L(M’)
- 假设 s 和 t 为 M 的两个状态，称 s 和 t 等价：如果从状态 s 出发能读出某个字α而停止于终态，那么同样，从 t 出发也能读出α而停止于终态；反之亦然
- 两个状态不等价，则称它们是可区别的

测试：状态的可区分性

- 两个状态 s 和 t 是可区分的，是指 ( )

A. 对于任意字α，要么 s 读出α停止于终态而 t 读出α停止于
终态而 s 读出

B. 存在一个字
t 读出α停止于
终态而 s 读出



DFA M 最少化的基本思想

- 把 M 的状态集划分为一些不相交的子集，使得任何两个不同子集的状态是可区别的，而同一子集的任何两个状态是等价的。最后，让每个子集选出一个代表，同时消去其他状态

测试：初始划分

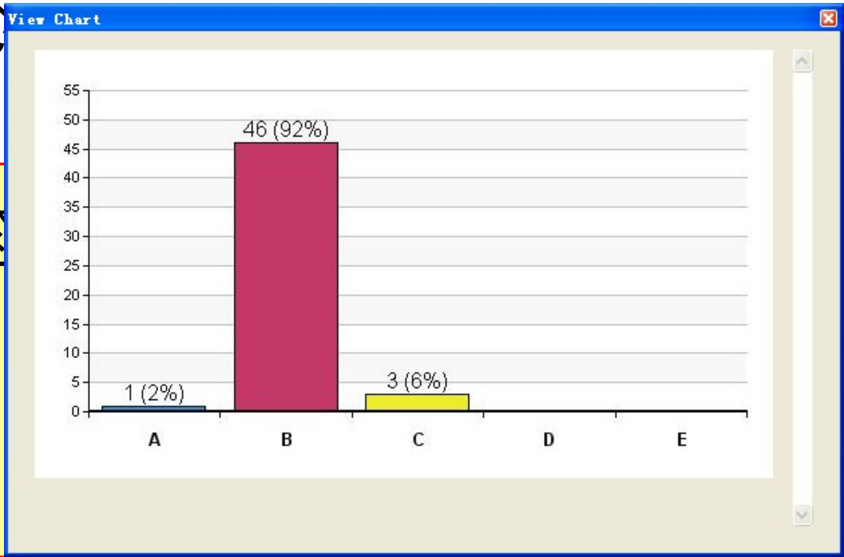
- 按照上述原则对 DFA 的状态集合 S 进行第一次划分，正确的分法是 ( )

A. 初态和非初态

B. 终态和非终态

C. 初态、终态、其他状态

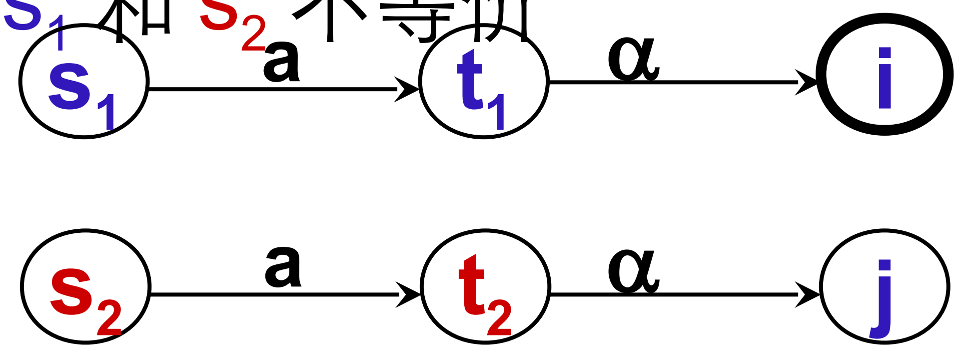
- 把 M 的状态
子集，使得
是可区别的
状态是等价

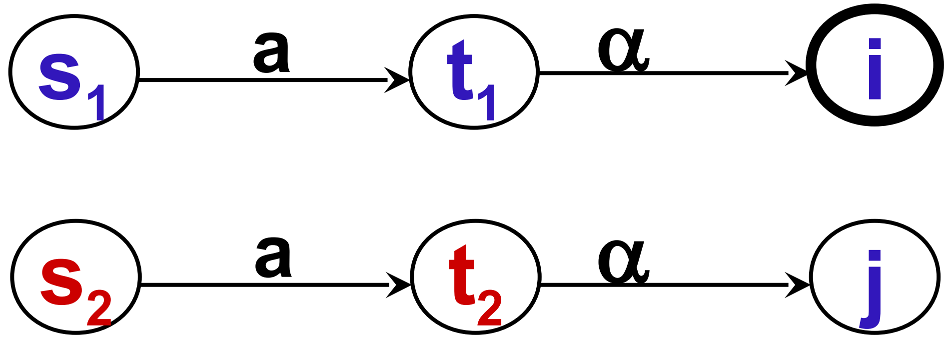


对 M 的状态集进行划分

- 首先，把 S 划分为终态和非终态两个子集，形成基本划分Π。
- 假定到某个时候，Π已含 m 个子集，记为 $\Pi=\{I^{(1)}, I^{(2)}, \ldots, I^{(m)}\}$，检查$\Pi$中的每个子集看是否能进一步划分：
  - 对某个 $I^{(i)}$，令 $I^{(i)}=\{s_1,s_2,\ldots,s_k\}$，若存在一个输入字符 a 使得 $I_a^{(i)}$ 不会包含在现行Π的某个子集 $I^{(j)}$ 中，则至少应把 $I^{(i)}$ 分为两个部分。

- 假定状态 s_1 和 s_2 经 a 弧分别到达 t_1 和 t_2
- t_1 和 t_2 属于现行Π中的两个不同子集
  - 说明有一个字α， t_1 读出α后到达终态，而 t_2 读出α后不能到达终态，或者反之
- 那么对于字 aα ， s_1 读出 aα 后到达终态，而 s_2 读出 aα 不能到达终态，或者反之
- 所以 s_1 和 s_2 不等价





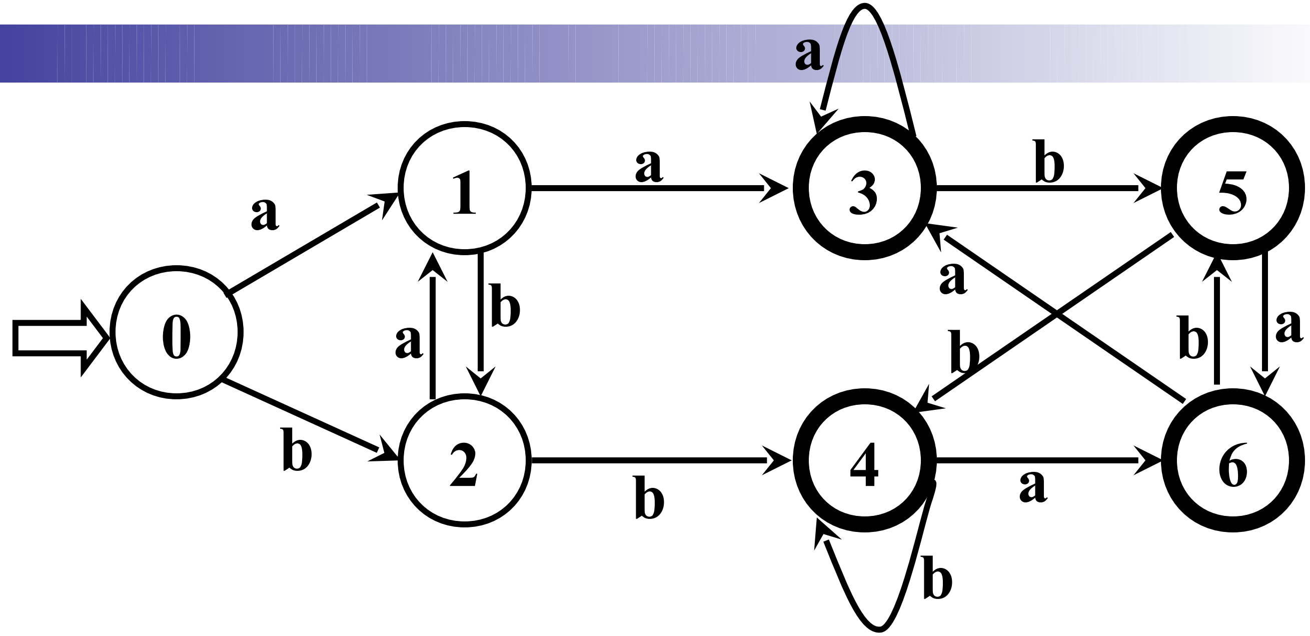
- 将 $I^{(i)}$ 分成两半，使得一半含有 s_1：

 $I^{(i1)}=\{s|s\in I^{(i)}$ 且 s 经 a 弧到达 t,

 且 t 与 t_1 属于现行Π中的同一子集}

 另一半含有 s_2：$I^{(i2)}=I^{(i)}-I^{(i1)}$

- 一般地，对某个 a 和 $I^{(i)}$，若 $I_a^{(i)}$ 落入现行Π中 N 个不同子集，则应把 $I^{(i)}$ 划分成 N 个不相交的组，使得每个组 J 的 J_a 都落入的Π同一子集。这样构成新的划分。
- 重复上述过程，直到Π所含子集数不再增长。
- 对于上述最后划分Π中的每个子集，我们选取每个子集 I 中的一个状态代表其他状态，则可得到化简后的 DFA M’ 。
- 若 I 含有原来的初态，则其代表为新的初态，若 I 含有原来的终态，则其代表为新的终态。



$I^{(1)}=\{0, 1, 2\}$ $I^{(2)}=\{3, 4, 5, 6\}$

$I_a^{(1)}=\{1, 3\}$

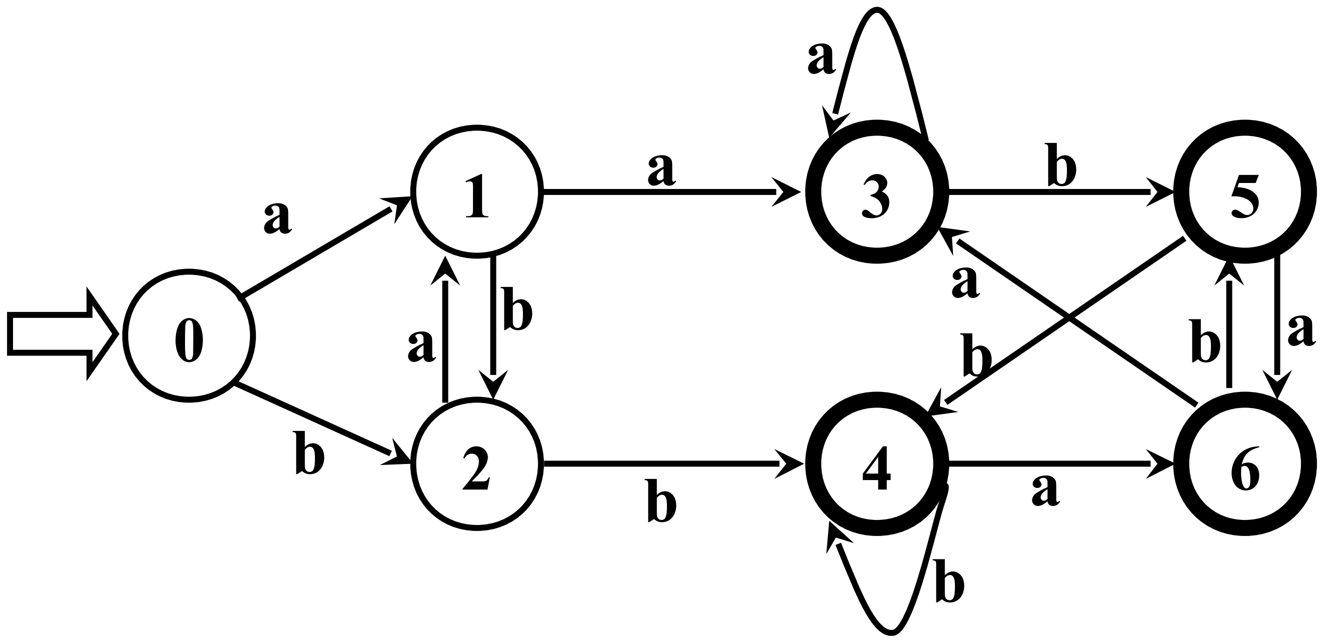
$I^{(11)}=\{0, 2\}$ $I^{(12)}=\{1\}$ $I^{(2)}=\{3, 4, 5, 6\}$

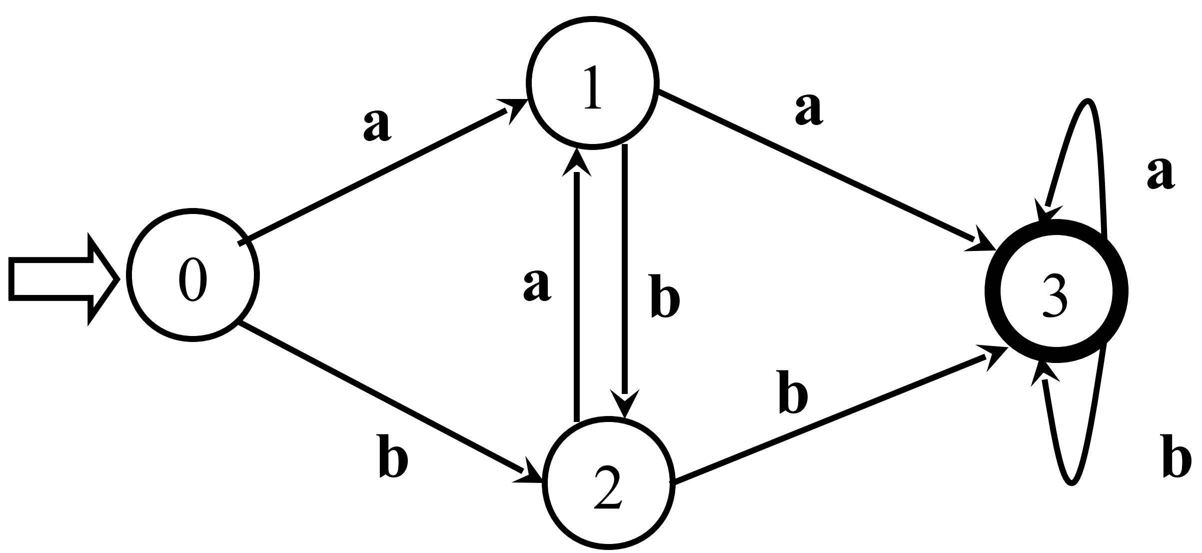
$I^{(11)}=\{0, 2\}$

$I_a^{(11)}=\{1\}$ $I_b^{(11)}=\{2, 4\}$

$I^{(111)}=\{0\}$ $I^{(112)}=\{2\}$ $I^{(12)}=\{1\}$ $I^{(2)}=\{3, 4, 5, 6\}$

$I_a^{(2)}=\{3, 6\}$ $I_b^{(2)}=\{4, 5\}$

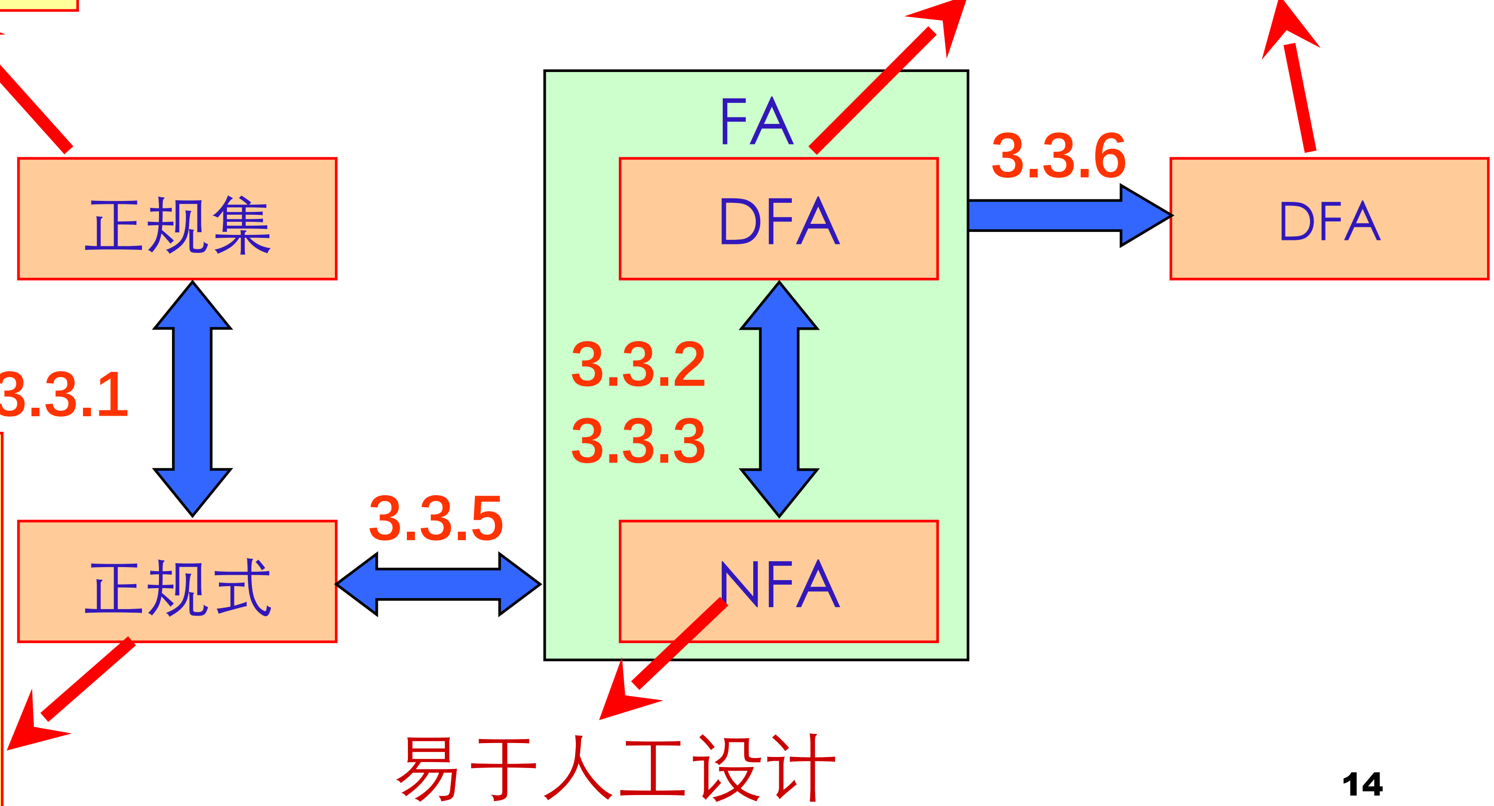

0
1
2
3
4
5
6
a
b


0
1
2
3
a
b

关系图

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then  返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169
…

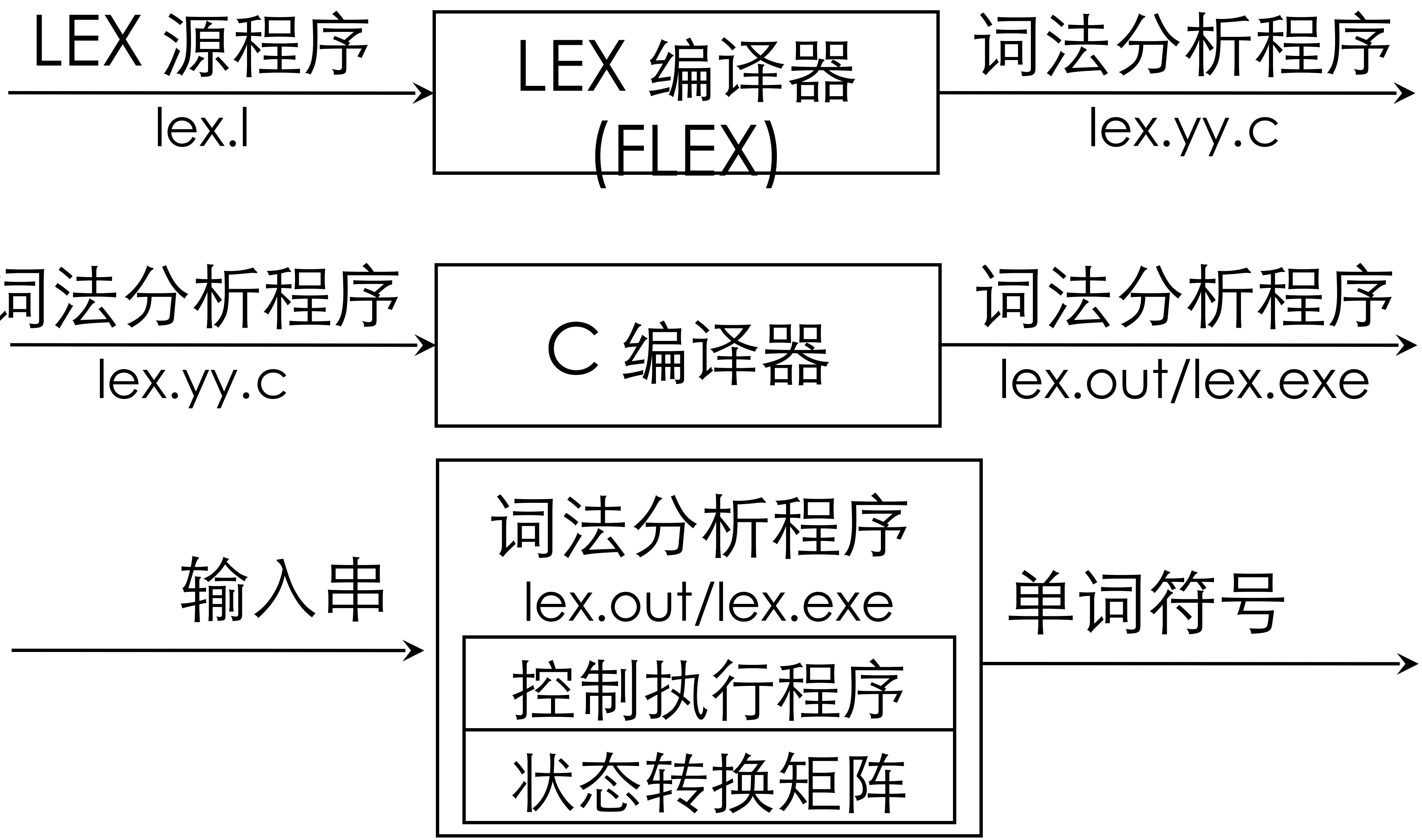


DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

3.4 词法分析器的自动产生 --LEX



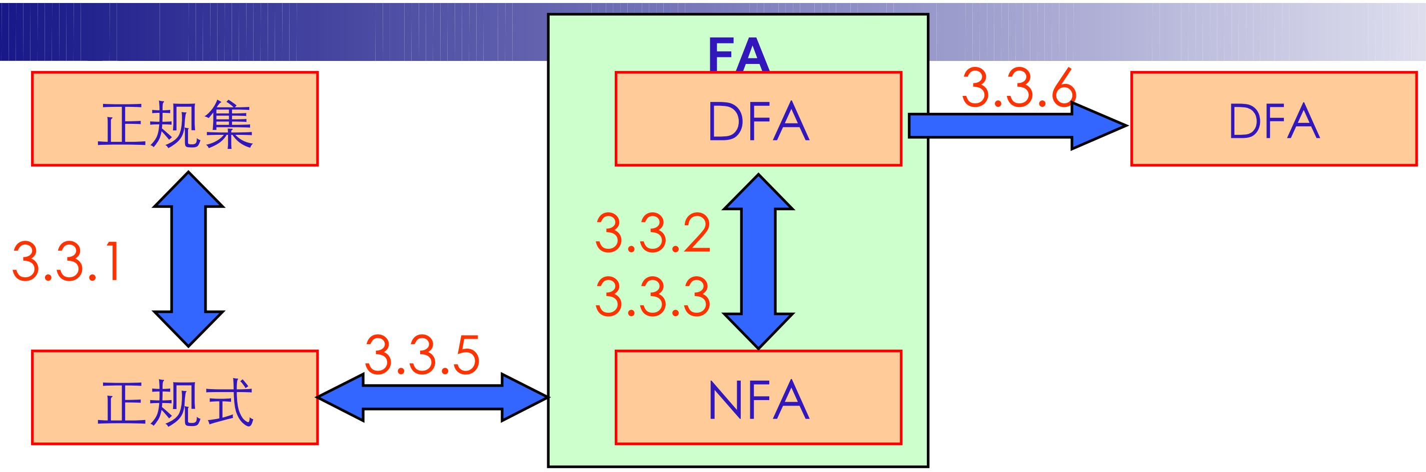
AUXILIARY DEFINITION

letter→A|B|...|Z
digit →0|1|...|9

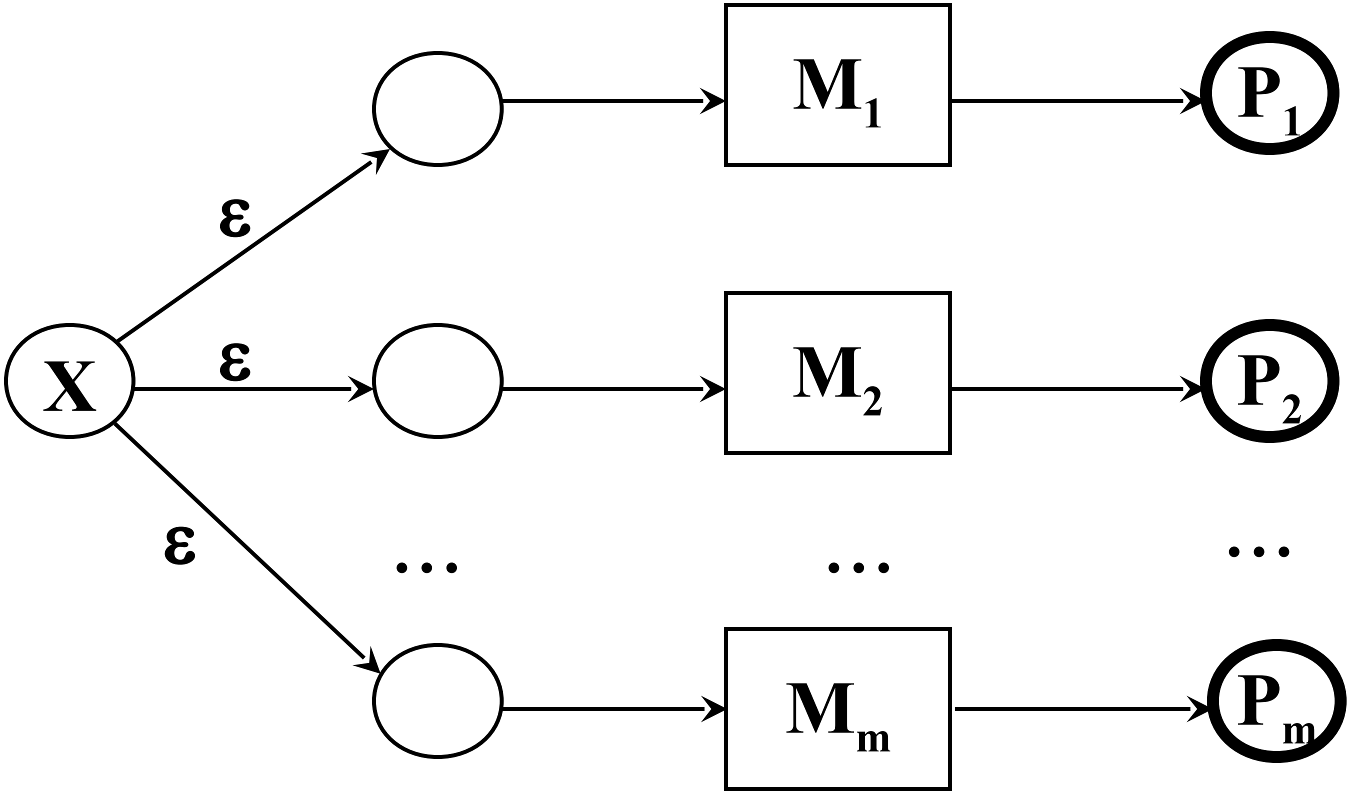
正规式

RECOGNITION RULES

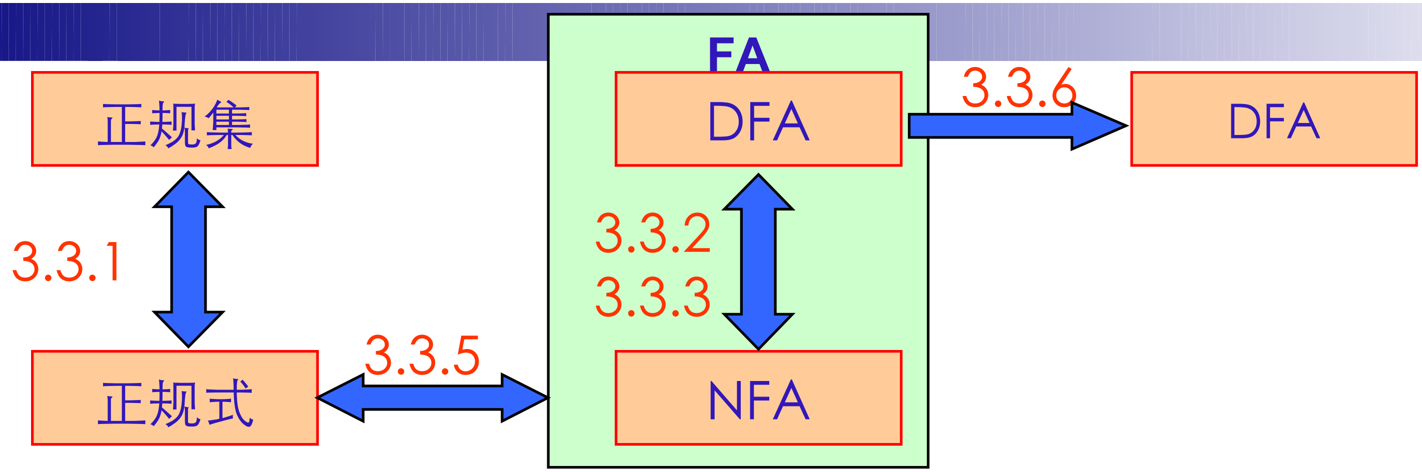
1	DIM	{ RETURN (1,-) }
2	IF	{ RETURN (2,-) }
3	DO	{ RETURN (3,-) }
4	STOP	{ RETURN (4,-) }
5	END	{ RETURN (5,-) }
6	letter(letter|digit) *	{ RETURN (6, TOKEN) }
7	digit(digit)*	{ RETURN (7, DTB) }
8	=	{ RETURN (8, -) }
9	+	{ RETURN (9,-) }
10	*	{ RETURN (10,-) }
11	**	{ RETURN (11,-) }
12	,	{ RETURN (12,-) }
13	(	{ RETURN (13,-) }
14	)	{ RETURN (14,-) }



- LEX 的工作过程：
  - 首先，对每条识别规则 P_i 构造一个相应的非确定有限自动机 M_i；
  - 然后，引进一个新初态 X，通过ε弧，将这些自动机连接成一个新的 NFA；



X
ε
ε
ε
M1
M2
Mm
P1
P2
Pm



- LEX 的工作过程：
  - 首先，对每条识别规则 P_i 构造一个相应的非确定有限自动机 M_i；
  - 然后，引进一个新初态 X，通过ε弧，将这些自动机连接成一个新的 NFA；
  - 最后，把 M 确定化、最小化，生成该 DFA 的状态转换表和控制执行程序

LEX 参考资料

- Yacc 与 Lex 快速入门
  - http://www.ibm.com/developerworks/cn/linux/sdk/lex/index
  - UNIX, LINUX
- The Lex & Yacc Page
  - http://dinosaur.compilertools.net/
- Flex (The Fast Lexical Analyzer)
  - http://flex.sourceforge.net/
  - for Windows: http://gnuwin32.sourceforge.net/packages/flex.htm

实验 :LEX(FLEX) 的使用

- 用 LEX 生成 PL 语言的词法分析器
  - 词法规则
    - 编译实习教材，表 17.2.1 PL 语言单词符号及其种别值
  - 功能
    - 输入一个 PL 语言源程序文件 demo.pl
    - 输出一个文件 tokens.txt，该文件包括每一个单词及其种别枚举值，每行一个单词
  - 提交 5 个文件
    - PL 语言的 LEX 源程序： pl.lex
    - PL 语言词法分析程序 C 源程序： lex.yy.c
    - PL 语言词法分析程序的可执行文件： pl.out/pl.exe
    - PL 语言源程序文件： demo.pl
    - 词法分析及结果文件： tokens.txt

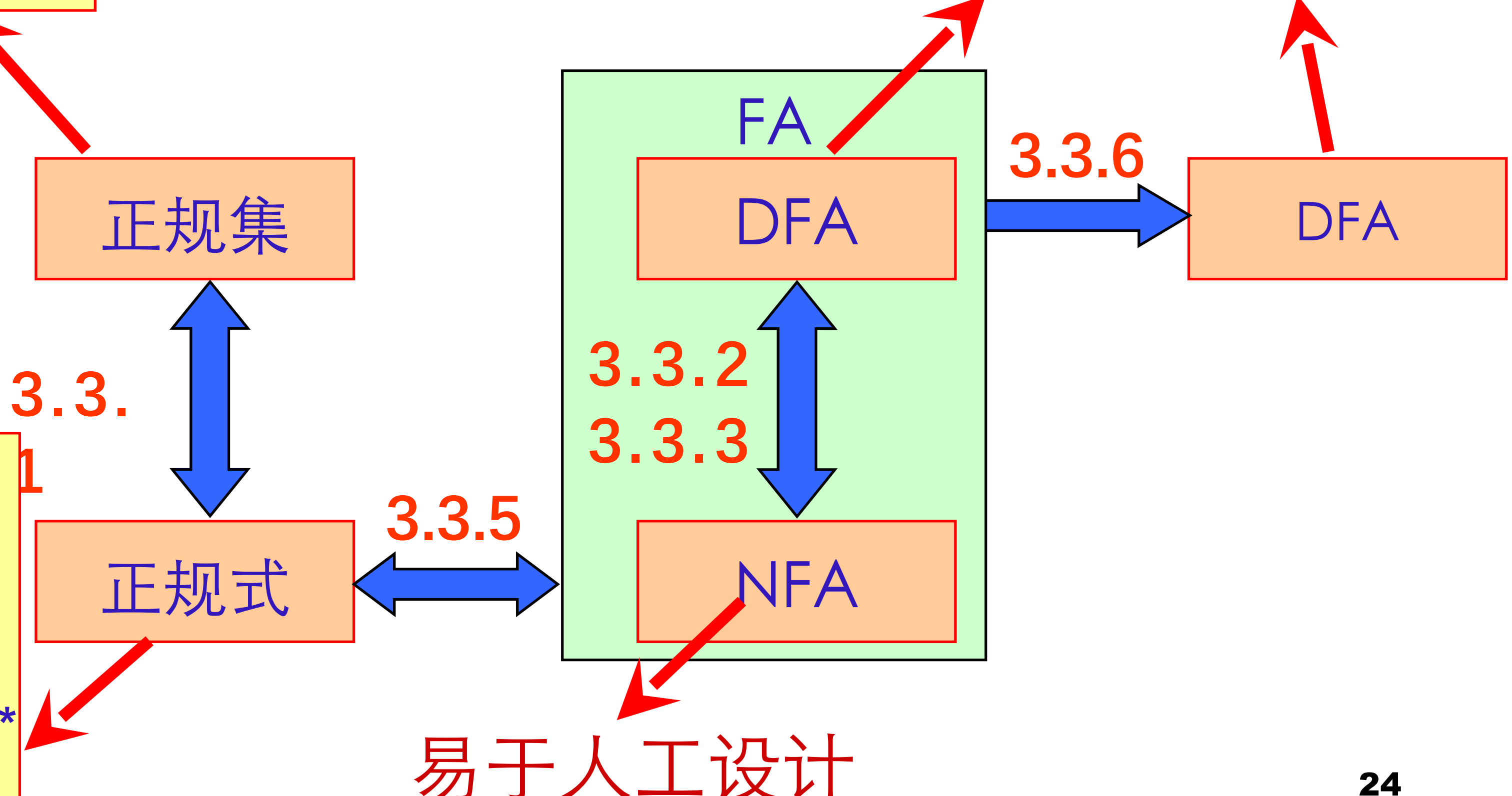
实验 :LEX(FLEX) 的使用

- 参考： Flex, version 2.5 文档
  - 阅读 (Flex for Windows 首页 .pdf)， 了解各压缩文件
  - 阅读 flex.pdf， 了解如何使用 Flex 及示例
    - 0.5 Some simple examples， scanner for a toy Pascal-like language

小结

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then  返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169
…



DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

编译原理

第三章 词法分析

第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

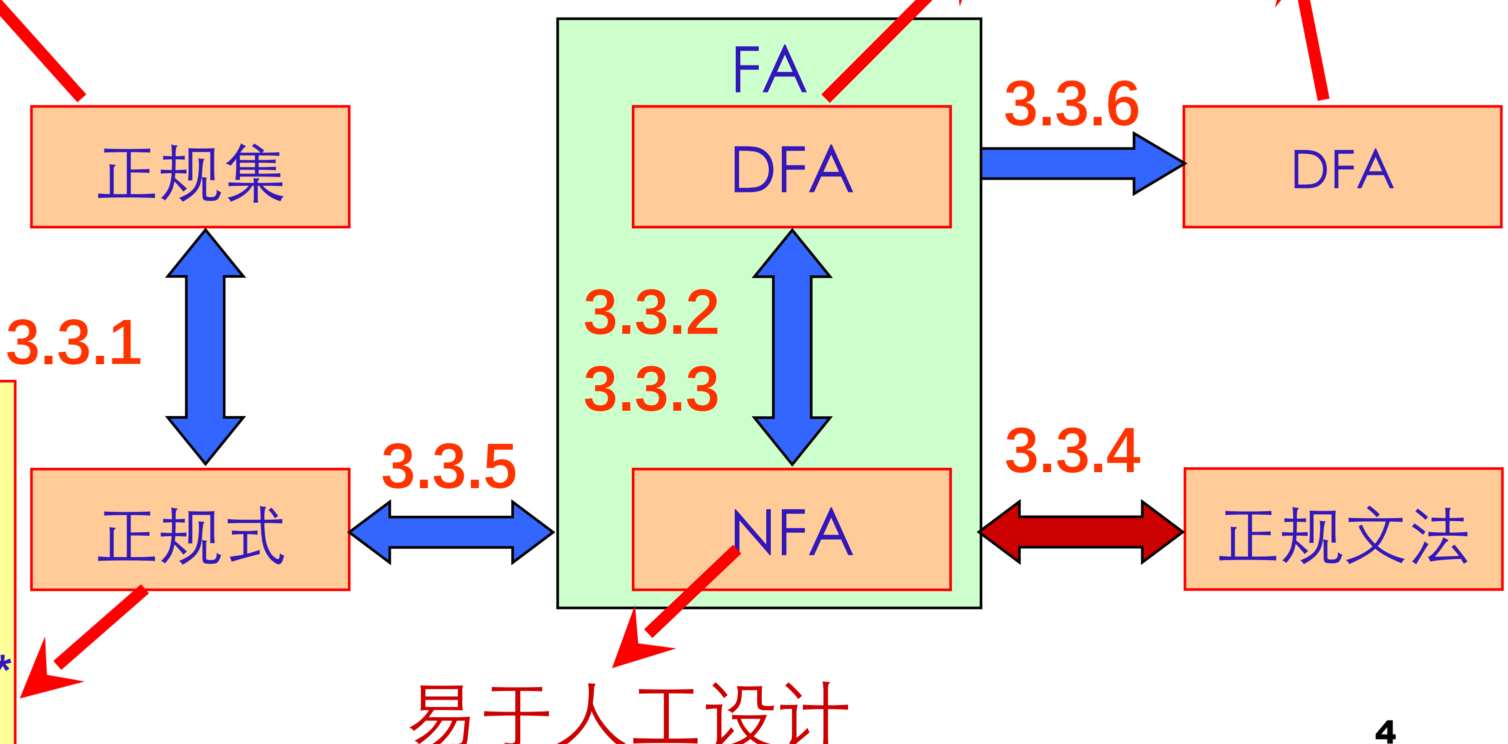
第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

关系图

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then  返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169
…



DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

形式语言鸟瞰

- 2 型 (上下文无关文法，非确定下推自动机)
  - 产生式形如： $A \rightarrow \beta$
  - 其中： $A \in V_N$; $\beta \in (V_T \cup V_N)^*$
- 3 型 (正规文法，有限自动机)
  - 产生式形如： $A \rightarrow \alpha B$ 或 $A \rightarrow \alpha$ **右线性文法**
  - 其中： $\alpha \in V_T^*$; A , $B \in V_N$
  - 产生式形如： $A \rightarrow B\alpha$ 或 $A \rightarrow \alpha$ **左线性文法**
  - 其中： $\alpha \in V_T^*$; A , $B \in V_N$

3.3.4 正规文法与有限自动机的等价性

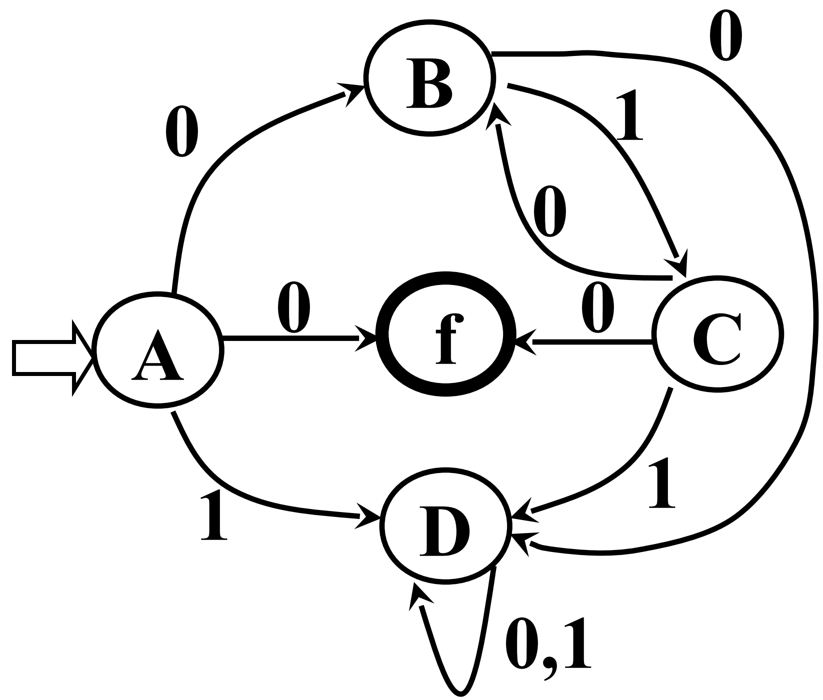
- 对于正规文法 G 和有限自动机 M，如果 L(G) ＝ L(M)，则称 G 和 M 是等价的
- 关于正规文法和有限自动机的等价性，有以下结论：

1. 对每一个右线性正规文法 G 或左线性正规文法 G，都存在一个有限自动机 (FA) M，使得 L(M) ＝ L(G)。
2. 对每一个 FA M，都存在一个右线性正规文法 G_R 和左线性正规文法 G_L，使得 $L(M) = L(G_R) = L(G_L)$。

$A \Rightarrow 0B$
$\Rightarrow 01C$
$\Rightarrow 010$

例：

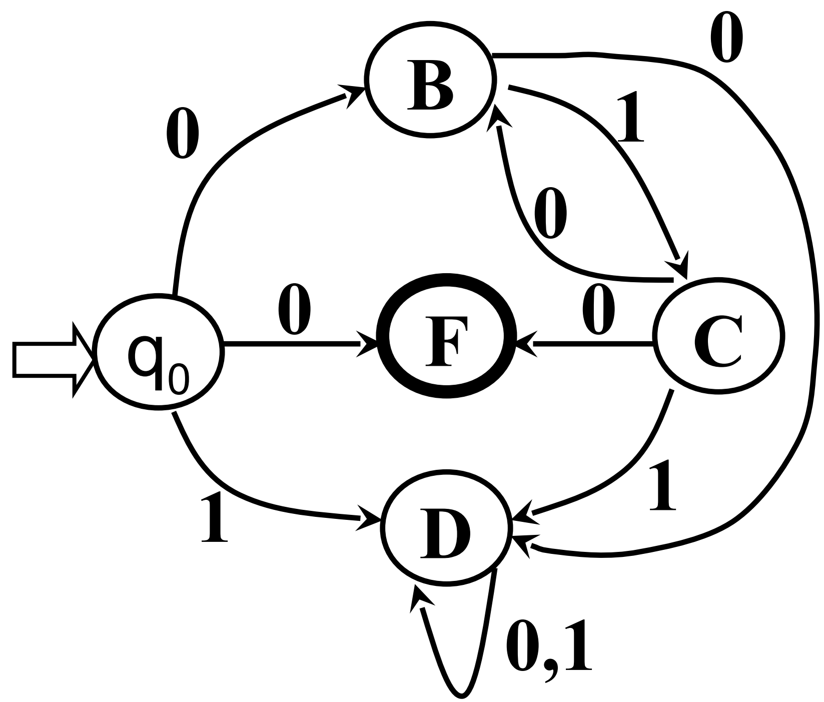
- $G_R(A)$ ：
  $A \to 0 \mid 0B \mid 1D$
  $B \to 0D \mid 1C$
  $C \to 0 \mid 0B \mid 1D$
  $D \to 0D \mid 1D$
- 从 G_R 出发构造 NFA M = <{A, B, C, D, f}, {0, 1}, δ', A, {f}>， M 的状态转换图如右图所示。
- 显然 $L(M) = L(G_R)$。



例：

$$F \Rightarrow C0 \Rightarrow B10 \Rightarrow 010$$

- 左线性正规文法 G_L = <{0, 1}, {B, C, D, F}, F, P′>，其中 P′ 由下列产生式组成：
  $F \to 0 \mid C0$
  $C \to B1$
  $B \to 0 \mid C0$
  $D \to 1 \mid C1 \mid D0 \mid D1 \mid B0$
- 从 G_L 出发构造 NFA M = <{q_0, B, C, D, F }, {0, 1}, δ, A, {F}>，M 的状态转换图如右图所示。
- 显然 $L(M) = L(G_L)$。



- 证明:

1. 对每一个右线性正规文法 G 或左线性正规文法 G，都构造一个有限自动机 (FA) M，使得 L(M) = L(G)。

   (1) 设右线性正规文法 $G=<V_T, V_N, S, P>$。将 V_N 中的每一非终结符号视为状态符号，并增加一个新的终结状态符号 f， $f \notin V_N$。

   令 $M=<V_N \cup \{f\}, V_T, \delta, S, \{f\}>$，其中状态转换函数δ由以下规则定义:

(a) 若对某个 $A \in V_N$ 及 $a \in V_T \cup \{\varepsilon\}$， P 中有产生式 $A \rightarrow a$，则令 $\delta(A,a)=f$

(b) 对任意的 $A \in V_N$ 及 $a \in V_T \cup \{\varepsilon\}$，设 P 中左端为 A，右端第一符号为 a 的所有产生式为：

$A \rightarrow aA_1|\cdots|aA_k$　　（不包括 $A \rightarrow a$），

则令 $\delta(A,a)=\{A_1,\cdots,A_k\}$。

显然，上述 M 是一个 NFA。

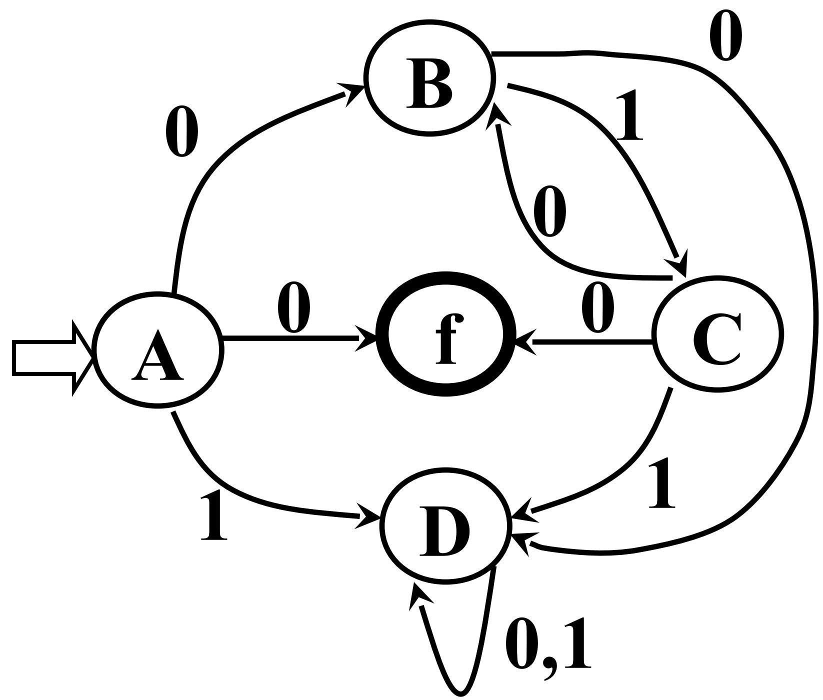
对于右线性正规文法 G，在 $S \overset{+}{\Rightarrow} w$ 的最左推导过程中：

- 利用 A→aB 一次就相当于在 M 中从状态 A 经过标记为 a 的箭弧到达状态 B （包括 a=ε 的情形）；
- 在推导的最后，利用 A→a 一次则相当于在 M 中从状态 A 经过标记为 a 的箭弧到达终结状态 f （包括 a=ε 的情形）。

综上，在正规文法 G 中，$S \overset{+}{\Rightarrow} w$ 的充要条件是：在 M 中，从状态 S 到状态 f 有一条通路，其上所有箭弧的标记符号依次连接起来恰好等于 w，这就是说，$w \in L(G)$ 当且仅当 $w \in L(M)$，故 $L(G) = L(M)$。

例：

- $G_R(A)$ ：
  $A \to 0 \mid 0B \mid 1D$
  $B \to 0D \mid 1C$
  $C \to 0 \mid 0B \mid 1D$
  $D \to 0D \mid 1D$
- 从 G_R 出发构造 NFA M = <{A, B, C, D, f}, {0, 1}, δ′, A, {f}> ， M 的状态转换图如右图所示。



- 显然 $L(M) = L(G_R)$ 。

3.3.4 正规文法与有限自动机的等价性

- 定理:

1. 对每一个右线性正规文法 G 或左线性正规文法 G，都存在一个有限自动机 (FA) M，使得 L(M) ＝ L(G)。

2. 对每一个 FA M，都存在一个右线性正规文法 G_R 和左线性正规文法 G_L，使得 $L(M) = L(G_R) = L(G_L)$。

(2) 设左线性正规文法 $G=<V_T, V_N, S, P>$ 。将 V_N 中的每一非终结符号视为状态符号，并增加一个初始状态符号 q_0 ， $q_0 \notin V_N$ 。

令 $M=<V_N \cup \{q_0\}, V_T, \delta, q_0, \{S\}>$ ，其中状态转换函数δ由以下规则定义：

(a) 若对某个 $A \in V_N$ 及 $a \in V_T \cup \{\varepsilon\}$ ，若 P 中有产生式 $A \to a$ ，则令 $\delta(q_0,a)=A$

(b) 对任意的 $A \in V_N$ 及 $a \in V_T \cup \{\varepsilon\}$ ，若 P 中所有右端第一符号为 A ，第二个符号为 a 的产生式为：

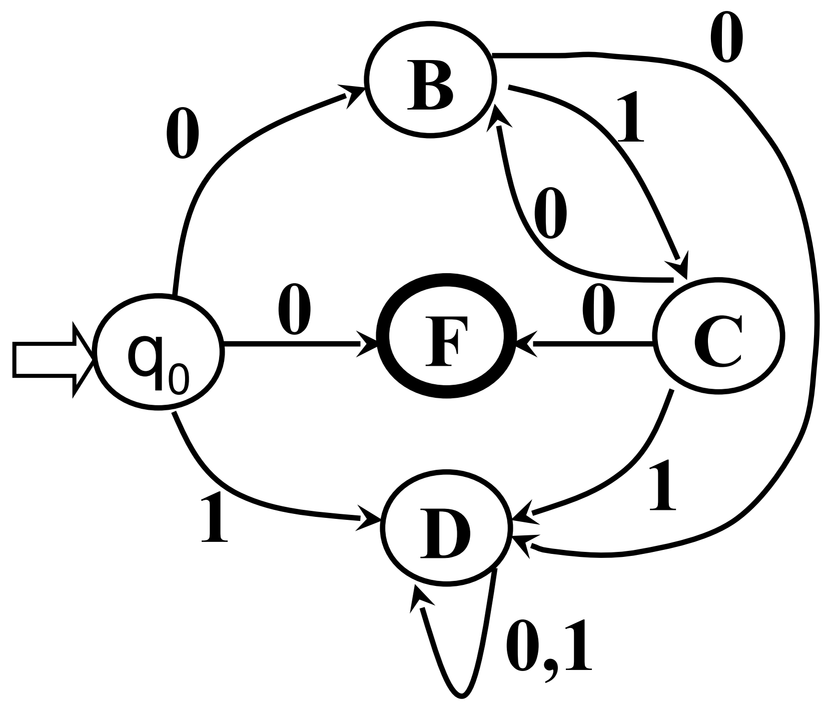
$$A_1 \to Aa, \cdots, A_k \to Aa ,$$

则令 $\delta(A,a)=\{A_1,\cdots,A_k\}$ 。

与(1)类似，可以证明 $L(G) = L(M)$

例：

- 左线性正规文法 G_L = <{0, 1}, {B, C, D, F}, F, P′>， 其中 P′ 由下列产生式组成：
  $F \to 0 \mid C0$
  $C \to B1$
  $B \to 0 \mid C0$
  $D \to 1 \mid C1 \mid D0 \mid D1 \mid B0$
- 从 G_L 出发构造 NFA M = <{q_0, B, C, D, F }, {0, 1}, δ, A, {F}> ， M 的状态转换图如右图所示。
- 显然 L(M) = L(G_L) 。



3.3.4 正规文法与有限自动机的等价性

- 定理:

  1. 对每一个右线性正规文法 G 或左线性正规文法 G，都存在一个有限自动机 (FA) M，使得 L(M) ＝ L(G)。

  2. 对每一个 FA M，都存在一个右线性正规文法 G_R 和左线性正规文法 G_L，使得 $L(M) = L(G_R) = L(G_L)$。

证明2：对每一个DFA M，都存在一个右线性正规文法 G_R 和左线性正规文法 G_L，使得 $L(M) = L(G_R) = L(G_L)$。

设DFA $M=<S, \Sigma, \delta, s_0, F>$

(1) 若 $s_0 \notin F$，我们令 $G_R=<\Sigma, S, s_0, P>$，其中P是由以下规则定义的产生式集合：

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

(a) 当 $B \notin F$ 时，令 $A \to aB$，

(b) 当 $B \in F$ 时，令 $A \to a|aB$。

对任何 $w\in\Sigma^*$，不妨设 $w=a_1\cdots a_k$，其中 $a_i\in\Sigma$ $(i=1,\cdots k)$。若 $s_0\overset{+}{\Rightarrow} w$，则存在一个最左推导：

$$s_0\Rightarrow a_1A_1\Rightarrow a_1a_2A_2\Rightarrow\cdots\Rightarrow a_1\cdots a_iA_i$$

$$\Rightarrow a_1\cdots a_{i+1}A_{i+1}\Rightarrow\cdots\Rightarrow a_1\cdots a_k$$

因而，在 M 中有一条从 s_0 出发依次经过 A_1，$\cdots$，A_{k-1} 到达终态的通路，该通路上所有箭弧的标记依次为 $a_1,\cdots,a_k$。反之亦然。所以，$w\in L(G_R)$ 当且仅当 $w\in L(M)$。

- 现在考虑 $s_0 \in F$ 的情形：

因为 $\delta(s_0, \varepsilon)=s_0$，所以 $\varepsilon \in L(M)$。但 ε 不属于上面构造的 G_R 所产生的语言 $L(G_R)$。不难发现，

$L(G_R)=L(M)-\{\varepsilon\}$。

所以，我们在上述 G_R 中添加新的非终结符号 s_0'，$(s_0' \notin S)$ 和产生式 $s_0' \rightarrow s_0 | \varepsilon$，并用 s_0' 代替 s_0 作

2. 对每一个 FA M，都存在一个右线性正规文法 G_R 和左线性正规文法 G_L，使得 $L(M) = L(G_R) = L(G_L)$。

(a) 当 $A= q_0$ 时，令 $B\rightarrow a$

,

(b) 当 $A\neq q_0$ 时，令

规文法

最后，

结论 2 得

3.3.4 正规文法与有限自动机的等价性

- 定理:

1. 对每一个右线性正规文法 G 或左线性正规文法 G，都存在一个有限自动机 (FA) M，使得 L(M) ＝ L(G) 。

2. 对每一个 FA M，都存在一个右线性正规文法 G_R 和左线性正规文法 G_L，使得 L(M) ＝ L(G_R) ＝ L(G_L) 。

例： 设 DFA M = <{A, B, C, D}, {0, 1}, δ, A, {B}> 。 M 的状态转换图如下图所示。

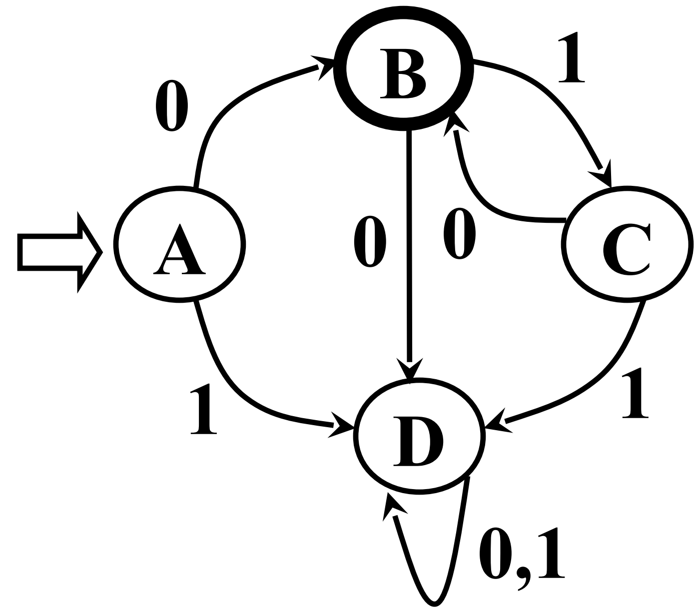
- L(M) = $0(10)^*$
- G_R = <{0, 1}, {A, B, C, D}, A, P> ， 其中 P 由下列产生式组成:

  A→0 | 0B | 1D

  B→0D | 1C

  C→0 | 0B | 1D

  D→0D | 1D

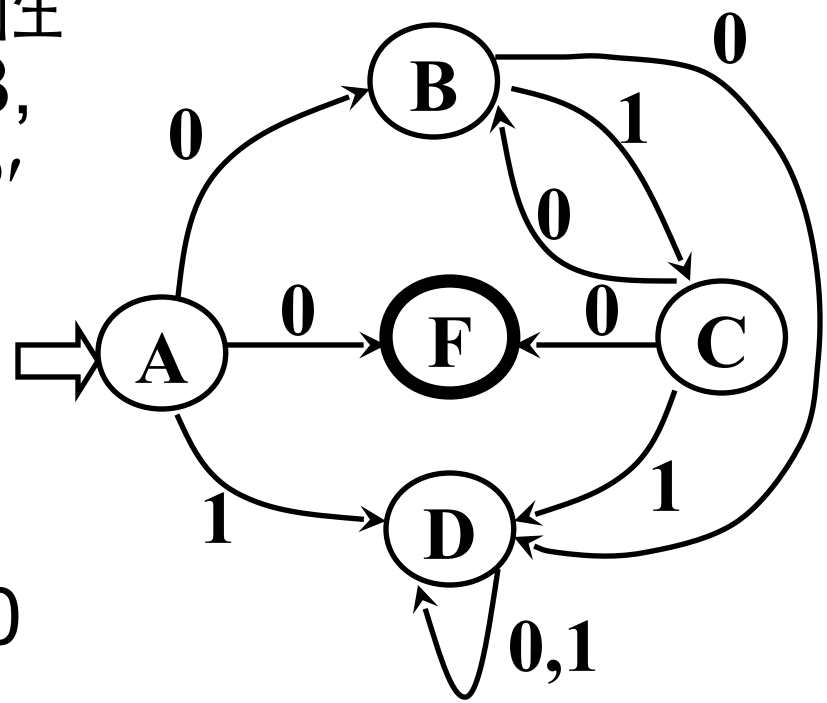


L(G_R) = L(M) = $0(10)^*$

例 设 DFA M = <{A, B, C, D, F }, {0, 1}, δ, A, {F}> 。 M 的状态转换图如下图所示。

- 从 NFA M 出发构造左线性正规文法 G_L = <{0, 1}, {B, C, D, F}, F, P′> ， 其中 P′ 由下列产生式组成：
  - F→0 | C0
  - C→B1
  - B→0 | C0
  - D→1 | C1 | D0 | D1 | B0

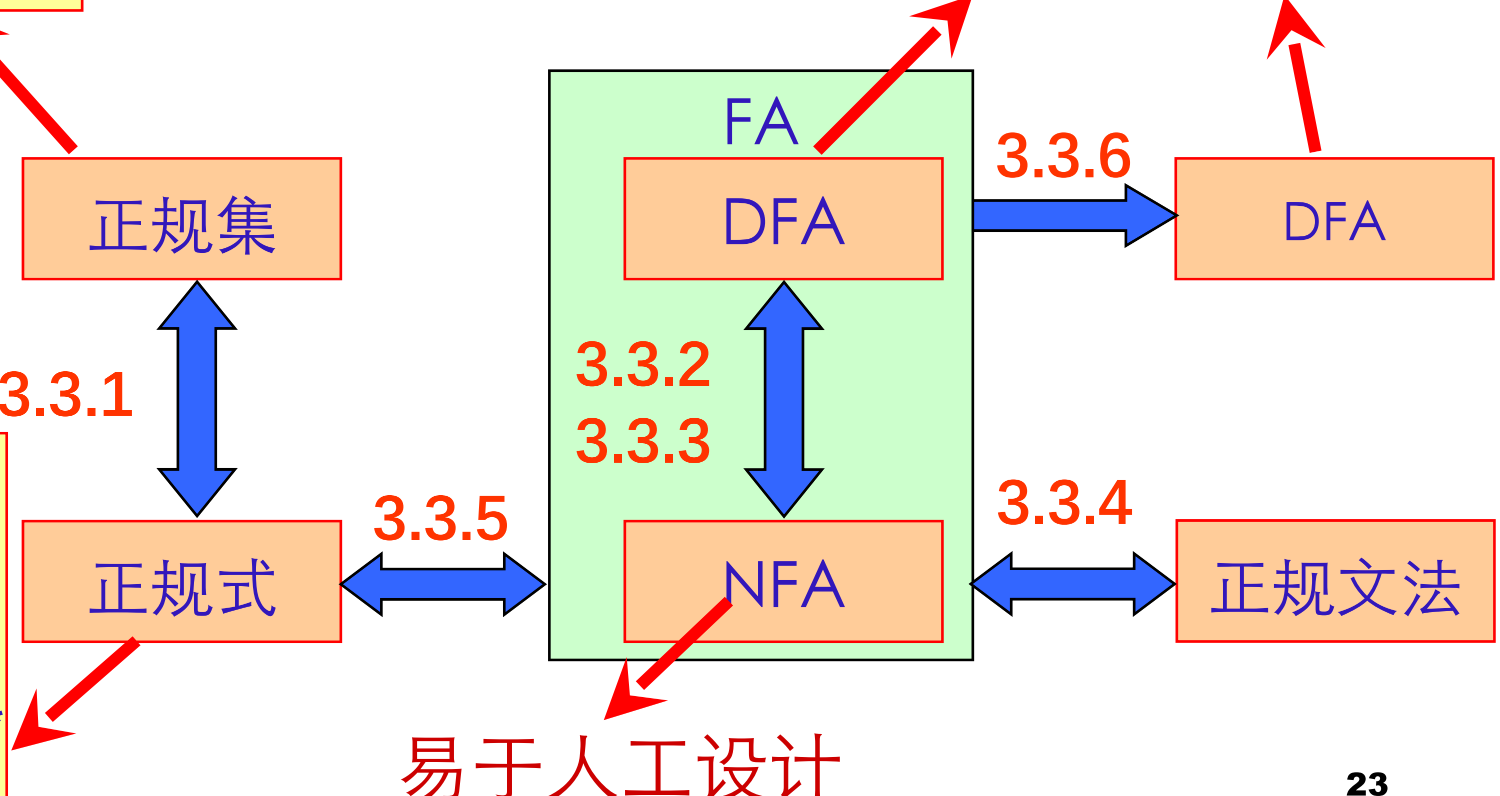
易证 $L(G_L) = L(M)$ 。



小结

```
curState = 初态
GetChar();
while( stateTrans[curState][ch] 有定义 ){
    // 存在后继状态，读入、拼接
    Concat();
    // 转换入下一状态，读入下一字符
   curState= stateTrans[curState][ch];
    if cur_state 是终态 then  返回 strToken 中的单
    GetChar( );
}
```

DIM,IF, DO,STOP,END
number, name, age
125, 2169
…



DIM
IF
DO
STOP
END
letter(letter|digit)*
digit(digit)*

第三章 词法分析

- 对于词法分析器的要求
- 词法分析器的设计
- 正规表达式与有限自动机
- 词法分析器的自动产生 --LEX

作业

- P64-7(选作 2 个小题)， 8(选作 3 个小题)， 12， 14