第 三 章

词 法 分 析

广东工业大学计算机学院

主观题 10分

设置

1. 写出生成语言$L=\{ a^n b^n c^m | n，m>=0\}$ 的上下文无关文法。

2. 令文法G［E］为：

Z→bMb

M→a|(L

L→Ma）

① 符号串b(Ma)b是否为该文法的一个句型，并证明。

② 若此符号串是句型，指出这个句型的所有短语、直接短语、句柄。

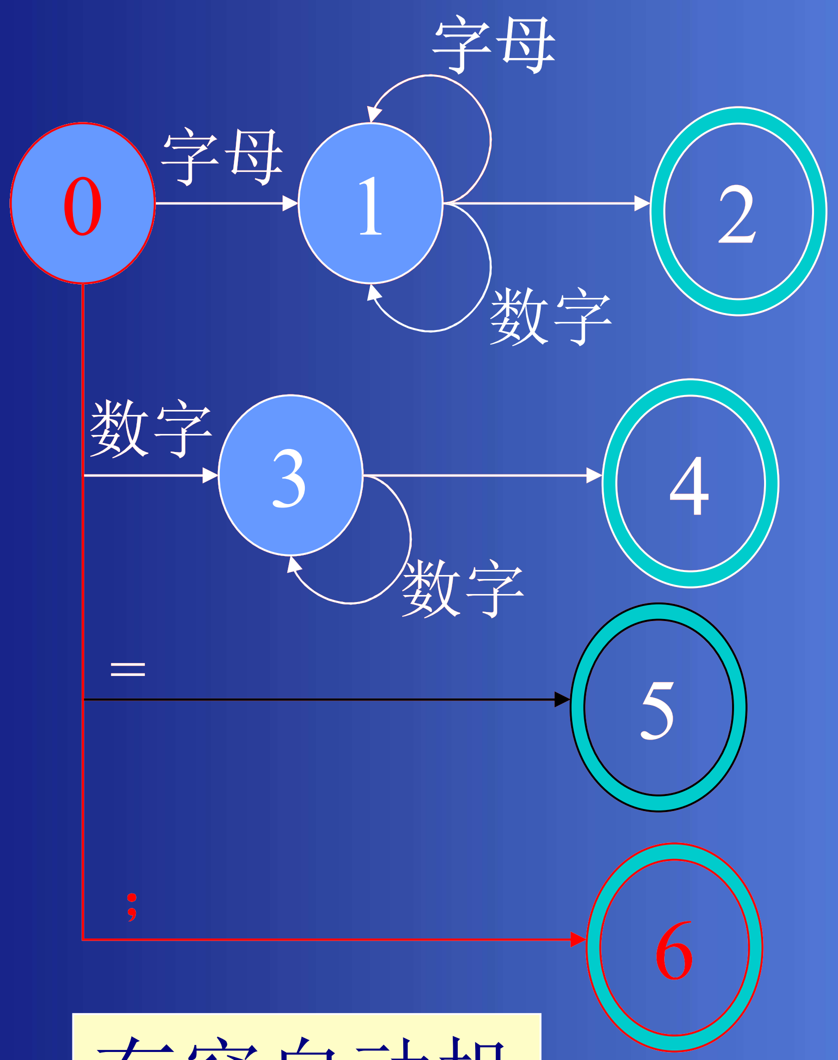
正常使用主观题需2.0以上版本雨课堂

作答

本章在编译程序中的地位

L	i	n	e	=	8	0	;

输入



输出

〈 id , ‘Line’〉
〈 = , —— 〉
〈 num, ‘80’〉
〈 ;, —— 〉

分析结束

有穷自动机

词法分析涉及的概念及关系

源程序(字符串) —扫描识别, 依据构词规则→ 单词符号串

词法分析程序(扫描器)

[输入]

输入, 扫描 预处理.

生成描述工具

[输出]

单词分类(3.1)

内部表示

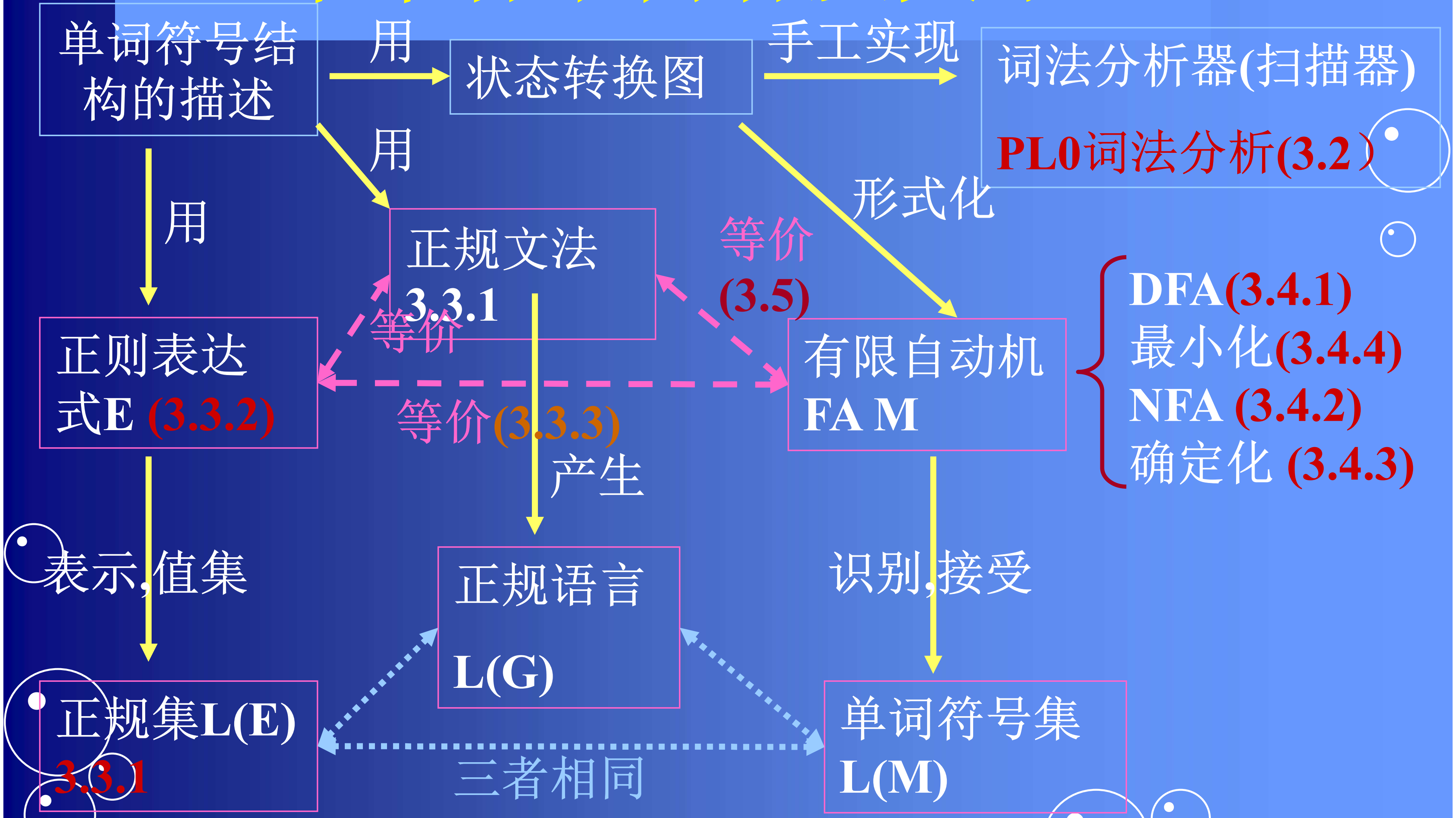
手工生成
- 状态转换图(3.4,3.5)
- 用程序实现, 即扫描器

自动生成等价
- 正规式3.3, 正规集
- 有限自动机(3.4,3.5,3.6)
  - DFA
  - NFA

  等价

自动生成工具LEX,用正规式描述, 扫描器象FA一样工作(3.7)

本章各节内容及关系



单词符号结构的描述
用
状态转换图
手工实现
词法分析器(扫描器)
PL0词法分析(3.2)
用
形式化
用
正规文法
3.3.1
等价
(3.5)
等价
正则表达式E (3.3.2)
等价(3.3.3)
有限自动机
FA M
DFA(3.4.1)
最小化(3.4.4)
NFA (3.4.2)
确定化 (3.4.3)
产生
表示,值集
识别,接受
正规语言
L(G)
正规集L(E)
3.3.1
单词符号集
L(M)
三者相同

本课内容

- 3.1 词法分析程序的设计（兼3.2）
- 3.3 单词的描述工具——正规式
- 3.4 有穷自动机(之DFA)

- 本章首先介绍词法分析程序的功能和设计原则，然后引入正规式和其对单词的描述，接着讲述有穷自动机理论，最后给出词法分析程序的自动构造原理。

3.1 词法分析程序

- 词法分析

  词法分析是逐个读入源程序字符，并按照构词规则分割成一系列单词，再转换成词标流的过程。单词是语言中具有独立意义的最小单位，词标是单词的机内表示，其格式由实现系统规定。

例如, **PL/0**语言的单词， 用户写的 **if a=2…**机内表示为**ifsym ident eql number…**

词法分析器的实现方式

- 词法分析是编译过程中的一个阶段，可以在语法分析前进行 。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。

①作为单独的一遍

字符序列　　　　单词序列

——>扫描器——>语法分析器——> . . .

②作为子程序

源程序 ——> 词法分析程序 ——Token——> 语法分析程序

词法分析程序 <——get token—— 语法分析程序

词法分析的任务

- 词法分析是编译的第一个阶段，它从左至右逐个字符地对源程序进行扫描，产生一个单词序列，用以语法分析。
- (1) 词法分析程序的主要任务：
  - 读源程序，产生单词符号
- (2) 词法分析程序的其它任务：
  - 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，……

单词符号的分类

- 单词符号是一个程序设计语言的基本语法符号。程序设计语言的单词符号一般可分成下列5种：
- ① 保留字，也称关键字，如PASCAL语言中的**begin, end, if, while**和**var**等。
- ② 标识符，用来表示各种名字，如常量名、变量名和过程名等。
- ③ 常数，各种类型的常数，如**25**，**3.1415**，**TRUE**和“**ABC**”等。
- ④ 运算符，如+，*，<=等。
- ⑤ 界符，如逗点，分号，括号等。

- 词法分析程序的功能是读入源程序，输出单词符号。

单词的表示

- 词法分析程序所输出的单词符号常常采用以下二元式表示：(单词种别，单词自身的值)。
  - 单词种别：语法分析需要的信息
  - 单词自身的值：编译其它阶段需要的信息。

比如在PASCAL的语句const i=25, yes=1；中的单词 25和1的种别都是常数，常数的值25和1对于代码生成来说，是必不可少的。

有时，对某些单词来说，不仅仅需要它的值，还需要其它一些信息以便编译的进行。

比如，对于标识符来说，还需要记载它的类别、层次还有其它属性，如果这些属性统统收集在符号表中，那么可以将单词的二元式表示设计成如下形式

(标识符，指向该标识符所在符号表中位置的指针)

如上述语句中的单词i和yes的表示为：

(标识符，指向i的表项的指针)

(标识符，指向yes的表项的指针)

实例

- 例如： 以下语句：
- **if i = 5 then**
- **x := y**

单词的种别可以用整数编码表示，假如

标识符编码为 **1**
常数为 **2**
保留字为 **3**
运算符为 **4**
界符为 **5**

- 经词法分析后所获得的单词为：

关键字**if** （**3**， ‘**if**’）
标识符**i** （**1**，指向**i**的符号表入口）
等号**=** （**4**， ‘**=**’）
常数**5** **(2, ‘5’)**
….
其它见书本**P38**

为何词法分析作为独立的阶段？

- 实际上，词法也是语法的一部分，词法描述完全可以归并到语法描述中去，只不过词法规则更简单些。
- 为什么把编译过程的分析工作划分成词法分析和语法分析两个阶段？主要的考虑因素为：
- ① 使编译程序的结构更简洁、清晰和条理化——词法分析更注重于处理源程序结构上的细节，如空格。
- ② 编译程序的效率会改进——可以建立词法分析的自动构造工具。
- ③ 增强编译程序的可移植性——例如处理与平台有关的字符。

单选题 2分

设置

此题未设置答案，请点击右侧设置按钮

词法分析器的输入是

A 单词符号

B 源程序

C .语法单位

D 目标程序

提交

单选题 2分

设置

此题未设置答案，请点击右侧设置按钮

词法分析器的输出结果是

A 单词的种别编码

B 单词在符号表中的位置

C 单词的种别编码和自身值

D 单词自身值

提交

PL/0编译程序的词法分析

- PL0单词的种类也有五种。（书P40）
  基本字：也可称为保留字或关键字，如**BEGIN**，**END**，**IF**，**THEN**等13个。
  运算符：如：+、-、*、／、：=、#、＞=、＜=等11个。
  标识符：用户定义的变量名、常数名、过程名。（1个，符合标识符定义即可）
  常数：如：10，25，100等整数。（无符号整数1个）
  界符：如：','、'.'、';'、'('、')'等5个。
- 如上，设计的单词符共有31个单词种别

PL/0编译程序的词法分析

- 全部单词种类由编译程序定义的纯量类型symbol给出，也可称为语法的词汇表。（书P40)
- PL/0编译程序中开始对类型的定义中给出“单词定义”为：

```
enum symbol{
    nul,  ident,  number,  plus,  minus,
    times,  slash,  oddsym,  eql,  neq,
lss,  leq,  gtr,  geq,  lparen,
rparen,  comma,  semicolon,period,  becomes,
beginsym, endsym, ifsym,  thensym,  whilesym,
writesym, readsym,  dosym,  callsym,  constsym,
varsym,  procsym,
};
#define symnum 32
```

关键字的处理

- PL/0编译程序文本中主程序开始对关键字表置初值如下（p431）：

关键字表char word[norw][al]为：

```
/*设置保留字名字,按照字母顺序,便于折半查找*/
strcpy(&(word[0][0]),"begin");
strcpy(&(word[1][0]),"call");
……
strcpy(&(word[11][0]),"while");
strcpy(&(word[12][0]),“write”);
```

查到时找到关键字相应的内部表示（enum symbol wsym[norw]）为：

```
wsym[0]=beginsym;
wsym[1]=callsym;
……
wsym[11]=whilesym;
wsym[12]=writesym;
```

运算符的内部表示

存放在enum symbol ssym[256]里，内部表示为：

```
ssym['+']=plus;
ssym['-']=minus;
ssym['*']=times;
ssym['/']=slash;
ssym['(']=lparen;
ssym[')']=rparen;
ssym['=']=eql;
ssym[',']=comma;
ssym['.']=period;
ssym['#']=neq;
ssym[';']=semicolon;
```

运算符的内部表示

PL0扫描下面语句

position := initial + rate * 60

则生成如下单词符号序列(单词种别序列）

ident becomes ident plus ident times number semicolon

问题：

那么**ident**对应的值，比如**position**怎么存在？

PL/0编译程序的词法分析

- 词法分析子程序
  - int getsym(

  PL/0编译程序设置了如下3个全程量的临时公用单元：
  - SYM：存放每个单词的类别(存放最近一次识别出来的token的类型)，用内部编码形式表示。
  - ID：存放用户所定义的标识符的值(存放最近一次识别出来的标识符的名字)
  - NUM：存放用户定义的数(存放最近一次识别出来的数字的值)。
- 功能
  - 从当前输入符号起扫描下一个词法单位，即单词
  - 通过下列全局变量传递单词的类别，用内部编码形式表示。
    - enum symbol sym;
  - 通过下列全局变量传递用户定义的标识符的值
    - char id [al+1]; /*al为标识符的最大长度*/
  - 通过下列全局变量传递常量单词的值
    - int num;

PL/0编译程序的词法分析

- 词法分析子程序getsym( ) 的处理流程
  - 从源程序扫描下一个字符(调用getch子程序)
  - -忽略空格、换行和TAB (每忽略一个，扫描下一个)
  - 识别单词(每扫描过一个字符，调用一次getch子程序)
    - 识别保留字

      设有一张保留字表。对每个字母打头后接字母或数字的字符串要查此表。若查着则为保留字，将对应的类别放在SYM中(类型enum symbol 见PPT第17）。如IF对应值IFSYM，THEN对应值为THENSYM。若查不着，则认为是用户定义的标识符。
    - 识别标识符

      对用户定义的标识符将IDENT放在SYM中，标识符本身的值放在ID中。

PL/0编译程序的词法分析

- 拼数

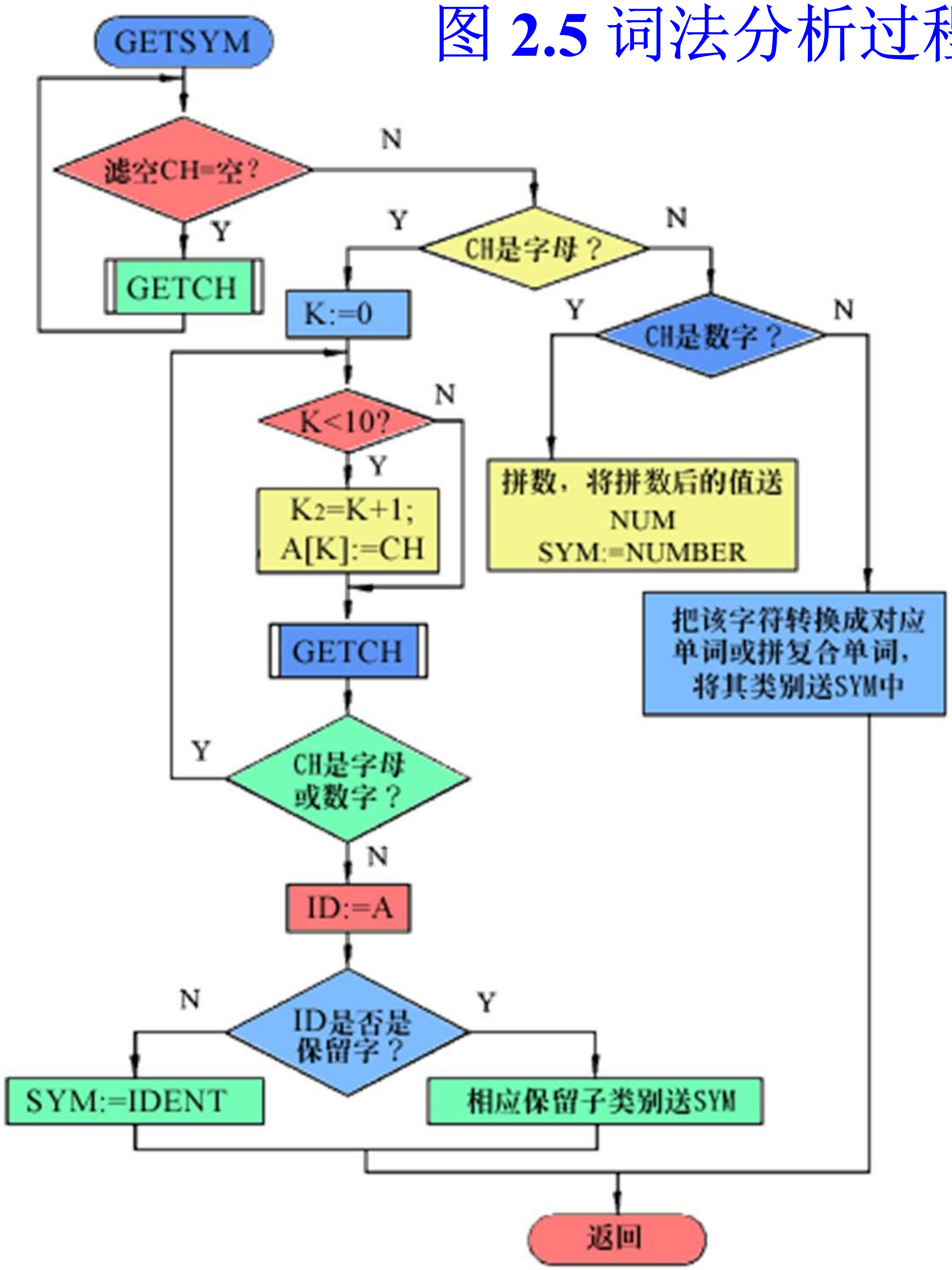
  当所取单词是数字时，将数的类别NUMBER放在SYM中，数值本身的值存放在NUM中。
- 识别单字符单词
- 拼双字符单词（复合词）

  对两个字符组成的算符

  如：>=、：=、<= 等单词，识别 后将类别送SYM中。

  – **根据单词类别设置sym，id 和num（GetSYM重要功能)**

图 2.5 词法分析过程GETSYM

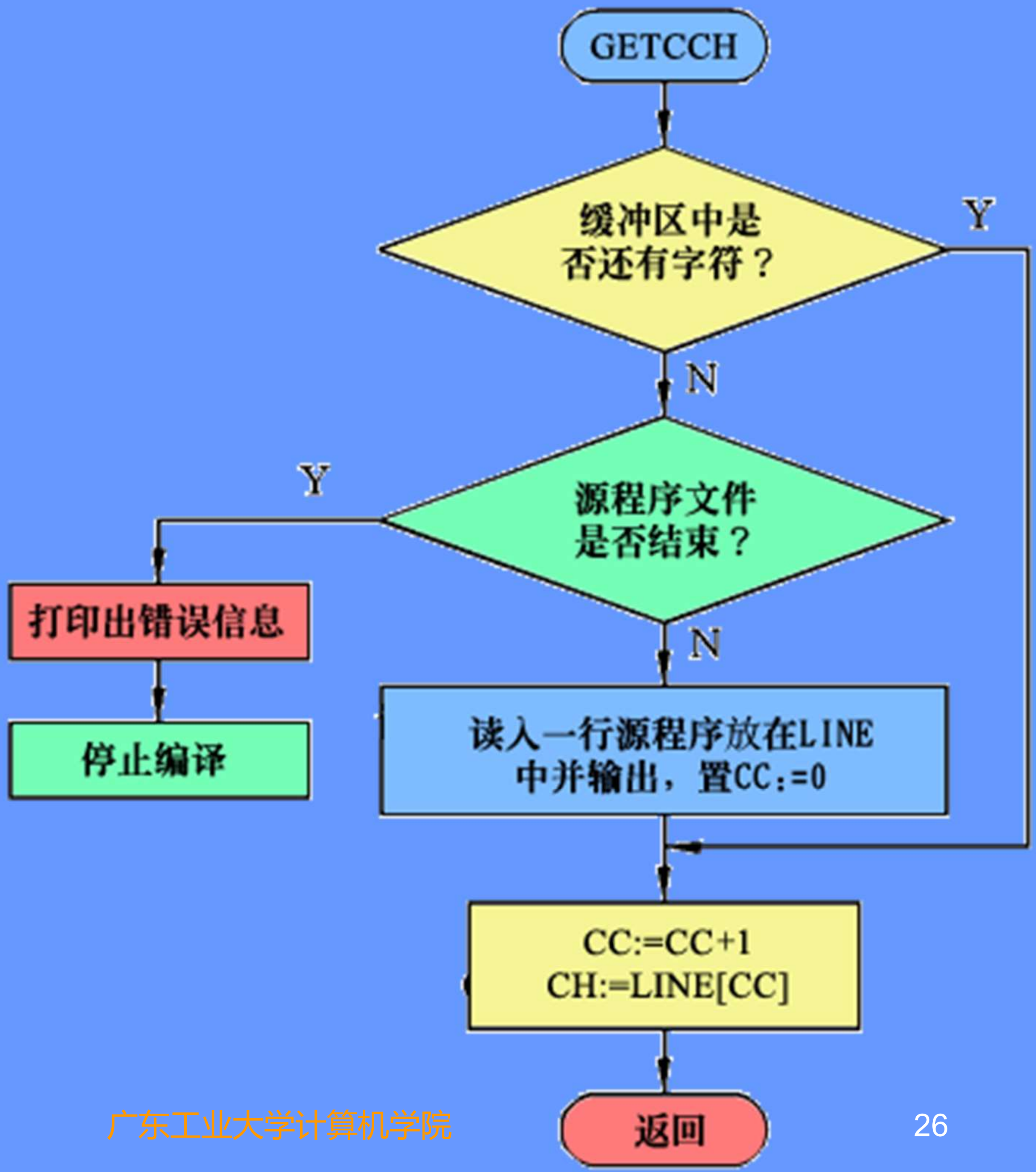


A：一维数组，数组元素为字符，最多个数为符号的最大长度10。

ID：同A。

word：保留字表

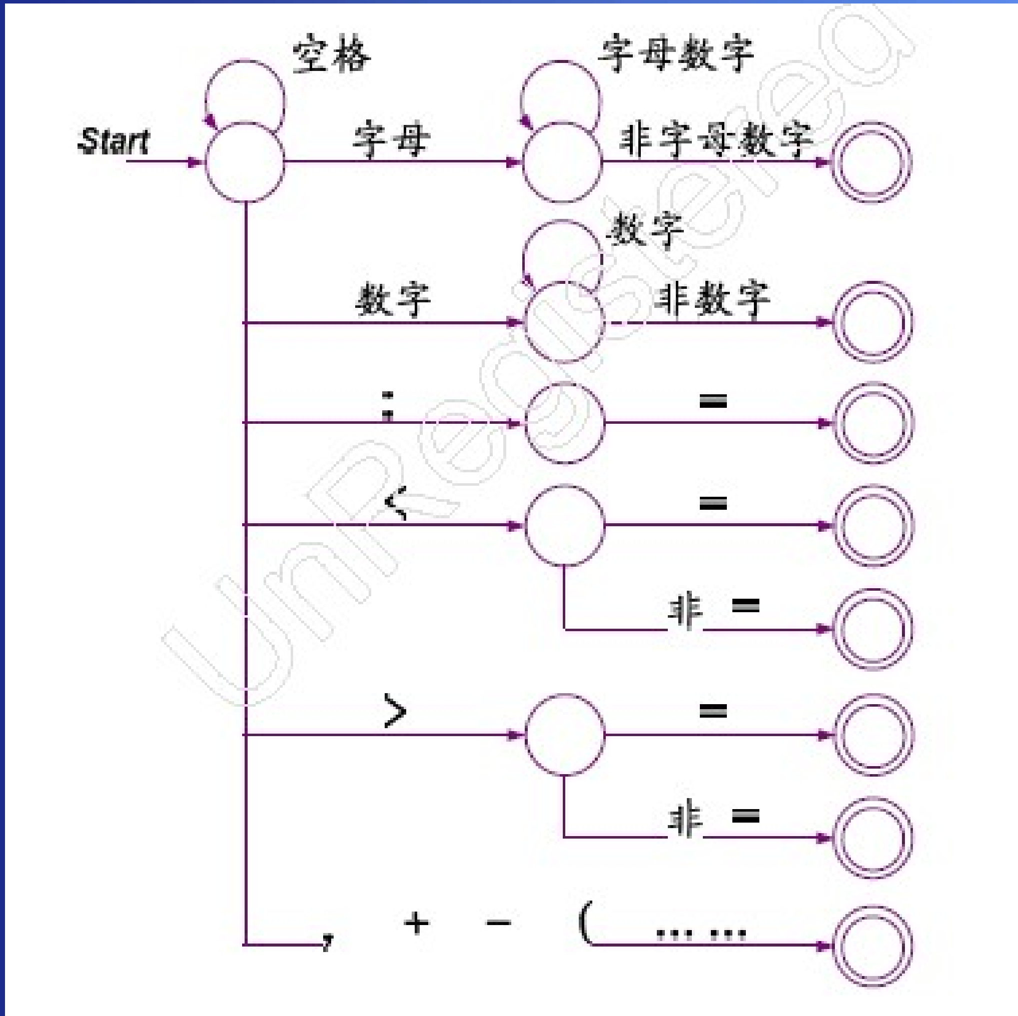
- **GETCH** 所用单元说明：
- CH： 存放当前读取的字符，初值为空。
- LINE： 为一维数组，其数组元素是字符，界对为1：80。用于读入一行字符的缓冲区。
- LL和CC为计数器，初值为0



PL0单词的识别

- - 可以借助于状态转换图进行设计
- - 状态转换图类似于有限状态自动机
- - 识别标识符单词、数字单词、单字符单词和双字符单词的状态转换图见下页
- - 保留字单词的识别可以在识别标识符单词的基础上，再去检索保留字表

实现单词识别的状态转换图



本课内容

- 3.1 词法分析程序的设计
- 3.3 单词的描述工具——正规式
- 3.4 有穷自动机(之DFA)

正规文法

- 多数程序设计语言的单词语法都能用正规文法（3型文法）来描述，下面先复习一下正规文法的定义：
- 设G = (V_n，V_T，P，S)，若P中的每一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，则G是3型文法或正规文法。
- 例如程序设计语言中的几类单词可用下列规则描述：
  <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> →d | d <无符号整数>
- <运算符> → + | - | * | / | =
- 其中l表示a~z中的任何一个英文字母；d表示0~9中的任一数字

正规式

- 正规(正则)表达式(**regular expression**)是说明单词的模式(**pattern**)的一种重要的表示法(记号)，是定义正规集的工具。
- 正规表达式可简称为正规式，其定义如下：
- 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{ }；
- ② 任何**a** ∈**Σ**，**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), $e_1 \mid e_2$, $e_1 \cdot e_2$, $e_1{}^*$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。

正规式的定义(续)

- 辅助字母表中各运算符的读法:
- (1) “|”读为“或”(也有使用“+”代替“|”的)
- (2) “·”读为“连接”; 如$e_1 \cdot e_2$,
- (3) “*”读为“闭包”(即，任意有限次的自重复连接)。
- (4) 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。
- 另外，“*”、“·”和“|”都是左结合的，表明表达式是自左至右求值。

正规式与正规集举例

- ∑= {a，b}，辅助字母表Σ’ = {Φ，ε，|，·，*，(，)}，则∑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)*　　　　　　{∑*上所有含有两个相继的a
- 　　　　　　或两个相继的b组成的串}

写正规表达式: 练习

给出下列在字母表{a,b}上的正规集的正规式

- 1. 以b开头，后跟若干个(至少1个) ab的符号串的集合。
- 2. 每个a都至少有一个b直接跟在其右边的符号串的集合。

- 1. 解： $bab(ab)^*$ 或 $b(ab)^+$
- 2. 解： $(b^*abb^*)^*$ 或 $(b^*ab^+)^*$

正规式满足的代数规律

- 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 / "或"的抽取律
- ⑤ $\varepsilon r = r, r\varepsilon = r$ ε是“连接”的恒等元素零一律
- ⑥ $r|r = r$
- ⑦ $r^* =\varepsilon|r|rr|\ldots$

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

正规式在程序设计语言中的应用

- 例1：令Σ= {l，d}，则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。
- 思考，如果用正规文法呢？
- 该例的正规式是：字母(字母|数字)*，表示的正规集中的每个元素的模式是“字母打头的字母数字串”，即Pascal和多数程序设计语言的标识符的词法规则。
- 例2：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。
- 例如2，12.59，3.6e2，471.88e-1都是该正规式的正规集中的元素。

正规文法和正规式的等价性

- 一个正规语言既可以由正规文法定义，也可以由正规式定义。
- 对于任意一个正规文法，存在一个定义同一个语言的正规式，
- 反之对于每个正规式，必定存在一个生成同一语言的正规文法。
- 所以正规文法和正规式之间存在等价性。
- 不过在实际使用中，有些正规语言很容易用文法定义，而有些正规语言则更容易用正规式定义。

正规式 ⇒ 正规文法

- 将∑上的一个正规式r转换成文法$G = (V_N, V_T, S, P)$。
- 令$V_T = \sum$，则应用如下规则，以产生正规文法中的规则：
- (1) 对形如$A \to x^*y$的正规式产生式，重写为：
  - $A \to xB$，其中B为一个新的非终结符
  - $A \to y$
  - $B \to xB$
  - $B \to y$
  - 实质：对*运算的变换

  （思考1：*xy时文法应该如何写？用上下文无关文法表示上述正规式呢？*

  思考2：为什么不写成$A \to xA|y$?）

- (2)对形如$A \to x \mid y$的正规式产生式，重写为：
  - $A \to x$
  - $A \to y$
  - 实质：对|运算的变换
- 不断应用上述规则做变换，知道每个产生式都符合正规文法的形式。

正规式 ⇒ 正规文法举例

- 例1：将r = a(a|d)*转换成正规文法。
- 解：step 1：
- S → a(a|d)* 替换为：
- S → aA　　A → (a|d)*
- step 2：
- A → (a|d)B　　A → ε
- step 3：
- B → (a|d)B　　B → ε
- step 4：
- S → aA
- A → aB　A → dB　A → ε
- B → aB　B → dB　B → ε

- 转换规则
- (1) A → x*y重写为：
  - ① A → xB，其中B为一个新的非终结符
  - ② A → y
  - ③ B → xB
  - ④ B → y
- (2) 对形如A → x | y的正规式产生式，重写为：
  - ⑤ A → x
  - ⑥ A → y

正规文法 ⇒ 正规式

- 将正规文法转换为正规式基本上是上述过程的逆过程，其转换规则如下：

	文法产生式	正规式
规则1	$A \to xB \quad B \to y$	$A = xy$
规则2	$A \to xA \mid y$	$A = x^*y$
规则3	$A \to x \quad A \to y$	$A = x \mid y$

正规文法 ⇒ 正规式举例

- 例2：设有文法G[S]
- $S \to aA$ $S \to a$ $A \to aA$ $A \to dA$ $A \to a$ $A \to d$
- 解：首先构造：
- $S = aA|a$ 【规则3】
- $A = (aA|dA)|(a|d)$ 【规则3】
- 再将A的正规式变换为：
- $A = (a|d)A|(a|d)$ 【外提A】
- 继续变换：$A = (a|d)^*(a|d)$ 【规则2】
- 再将A右端代入S的正规式得：
- $S = a(a|d)^*(a|d)|a$
- 再作正规式的代数变换：
- $S = a((a|d)^*(a|d)|\varepsilon)$ 【外提a】
- $S = a(a|d)^*$ 【利用ε化简】
- 即$a(a|d)^*$为与文法G[S]对应的正规式

	文法产生式	正规式
规则1	$A \to xB$ $B \to y$	$A = xy$
规则2	$A \to xA\|y$	$A = x^*y$
规则3	$A \to x$ $A \to y$	$A = x\|y$

单选题 2分

设置

此题未设置答案，请点击右侧设置按钮

编译技术中描述单词符号的形成规则的常用工具有

A 正规文法

B 正规式

C 有穷自动机

D 以上都是

提交

本课内容

- **4.1** 词法分析程序的设计
- **4.2** 单词的描述工具——正规式
- **4.3** 有穷自动机(之**DFA**)

有穷自动机的定义及分类

- 有穷自动机(也称有限自动机)是一种识别装置，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，目的是为词法分析程序的自动构造寻找特殊的方法和工具。

- 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 (Deterministic Finite Automata)**
  - 不确定的有穷自动机**NFA (Nondeterministic Finite Automata)**
  - **NFA**的确定化
  - **DFA**的最小化

1. 确定的有穷自动机DFA

- **DFA**定义：一个确定的有穷自动机**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；
- ③ **f**是转换函数，是**K×Σ→K**上的映射，即如果有$f(k_i, a) = k_j$，($k_i \in K$，$k_j \in K$)就意味着：当前状态为k_i，输入符号为**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 ∈ K**是唯一的一个初态；
- ⑤ **Z ⊂ K**是一个终态集，终态也称可接受状态或结束状态。

DFA M的确定性表现在映射**δ: S×∑→S**是一个单值函数。

单选题 2分

设置

此题未设置答案，请点击右侧设置按钮

____不是DFA的成分

A 有穷字母表

B 初态集合

C 终态集合

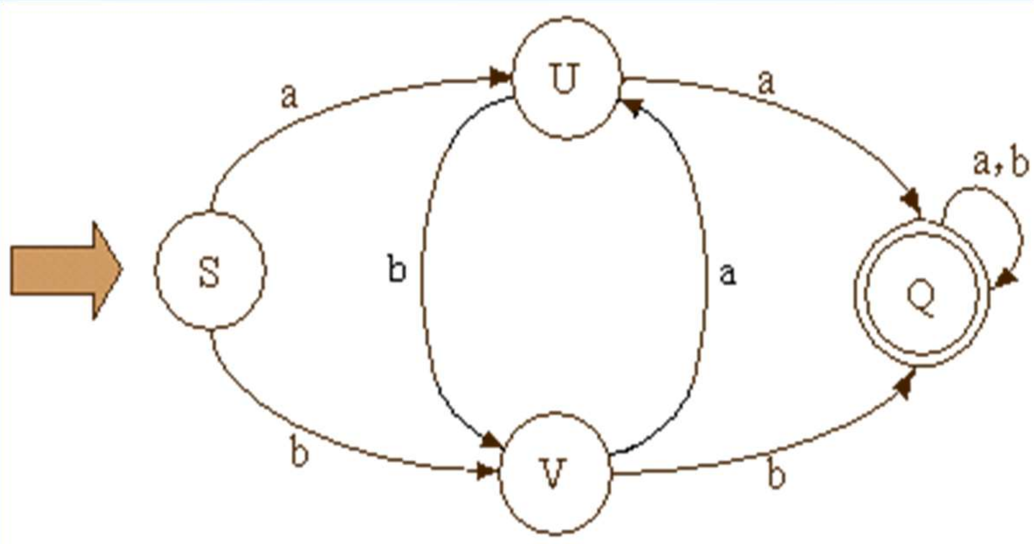
D 有穷状态集合

提交

DFA的状态图表示举例

- 设有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可以表示成状态图(或称状态转换图)的形式：
- (1) 结点对应于状态，弧对应于转换关系。
- (2) 初态结点冠以箭头“=>”或标以“-”，
- (3) 终态结点用双圈表示或标以“+”，

DFA的矩阵表示举例

- 设有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 符号	a	b	状态标志
S	U	V	0
U	Q	V	0
V	U	Q	0
Q	Q	Q	1

f是转换函数，是K×Σ→K上的映射

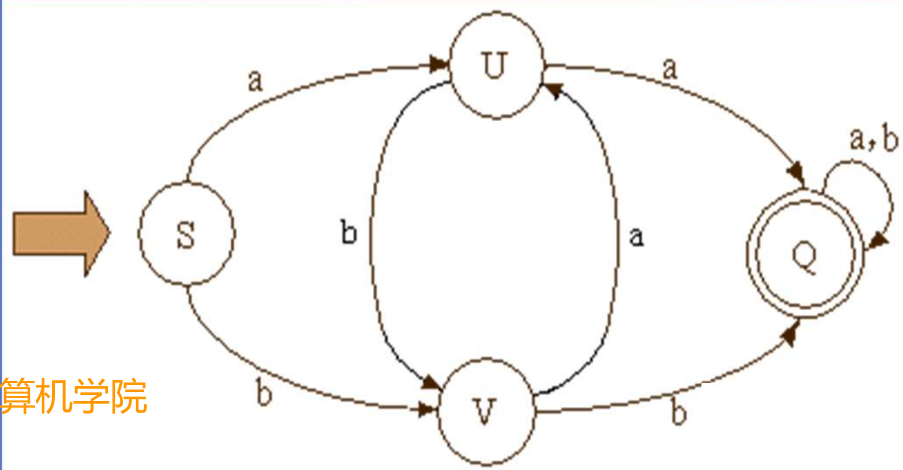
- DFA还可以表示成矩阵的形式：
  - 行：状态
  - 列：输入符号
  - 矩阵元素：将转换成的新状态。
  - “⇒”：表明初始状态，否则第一行即是初态
  - 状态标志：0表示非终端状态；1表示终端状态。

扩充转换函数

- 为描述一个符号串t可为DFA M所接受，需要将转换函数f作扩充：
- (1) 是$K \times \Sigma^* \to K$的映射，且设$Q \in K$，函数$f(Q, \varepsilon) = Q$，即：如果输入符号串始空串，则停留在原来的状态上；
- (2) 将输入符号串t表示成$t_1 t_x$，其中$t_1 \in \Sigma$，$t_x \in \Sigma^*$，在DFA M上运行的定义为：

$$f(Q, t_1 t_x) = f(\underline{f(Q, t_1)}, t_x)$$

- 例如：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



在DFA上接受符号串

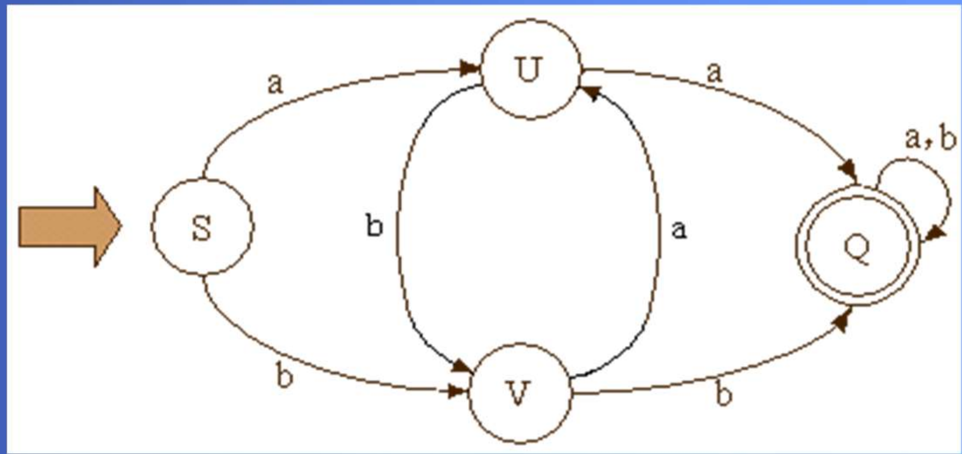
- 对DFA N = (K，∑，f，S，Z)有如下定义：
- **1. ∑*上的符号串t在DFA N上运行**
- 一个输入符号串t∈∑*，将t表示成t_1t_x，其中t_1∈∑，t_x∈∑*，如果

$$f(Q, t_1t_x) = f(f(Q, t_1), t_x)，其中Q\in K$$

- 则称t在DFA N上运行
- **2. ∑*上的符号串t被DFA N接受**
- 在1成立的基础上，若t∈∑*，f(S，t)=P，其中S为N的开始状态，P∈Z，Z为终态集，则称t为DFA N所接受(识别)。
- 注意：DFA N所能接受的符号串的全体记为L(N)，它是一个正规集。

在DFA上接受符号串举例

- 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



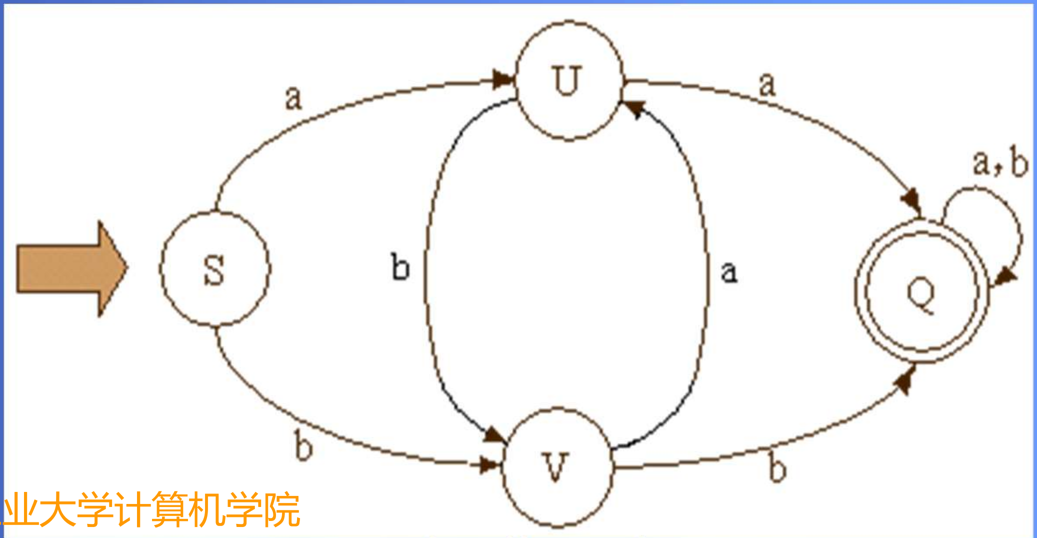
- 特别地, 若DFA M的始态结点又是终态结点，则ε为DFA M所接受。

DFA有关的结论

- L(M)：DFA M上所有能够接受的字符串的集合。
- DFA有关的结论：∑上的一个符号串集V⊂∑*是正规的，当且仅当存在一个∑上的DFA M，使得V = L(M)。
- 联系实际例子来说：所有像baab这样能被DFA M接受的符号串的集合，就是正规集。

- DFA的确定性表现在转换函数f：K×Σ→K是一个单值函数，即，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Σ含有n个输入字符，则在任意一个状态结点上最多有n条弧射出，而且每条弧以不同的输入字符标记。



DFA的程序模拟

- DFA的行为很容易用程序来模拟，这是DFA 得到广泛应用的重要原因。
- 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')
```

DFA的程序

- DFA与程序结构之间存在下述对应关系，并可以据此构造相应的程序：
  - ①初态对应程序的开始；
  - ②终态对应程序的结束，一般是一条返回语句，且不同的终态对应不同的返回语句；
  - ③状态转移分叉对应分情况或者条件语句 ；
  - ④转换图中的环对应程序中的循环语句;

作业

- 在本章最后一起布置。