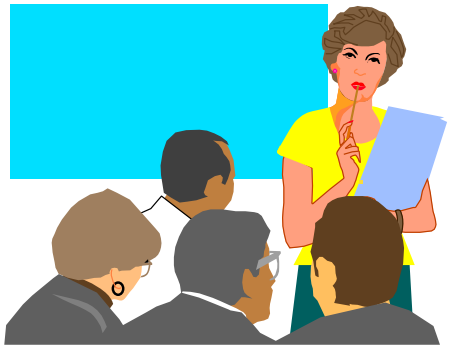
编译技术



胡春明
hucm@buaa.edu.cn

2018.9-2019.1

词法分析的功能

3.1 词法分析程序的功能及实现方案

☙ 词法分析程序的功能

- ♦ 词法分析：根据词法规则识别及组合单词，进行词法检查。
- ♦ 对数字常数完成数字字符串到二进制数值的转换。
- ♦ 删去空格字符和注释。

- ♦ 删去空格字符和注释。

词法分析

任务：依据文法（词法）分析和识别单词。

源程序是由字符序列构成的，词法分析扫描源程序(字符串),根据语言的词法规则分析并识别单词，并以某种编码形式输出。

•单词：是语言的基本语法单位，一般语言有四大类单词

<1>语言定义的关键字或保留字（如BEGIN、END、IF）

对于如下的字符串,词法分析程序将分析和识别出9个单词：

X1	:=	(	2.0	+	0.8	)	*	C1
1	2	3	4	5	6	7	8	9

3.2 单词的种类及词法分析程序的输出形式

单词的种类

1. 保留字：begin、end、for、do...
2. 标识符：由用户定义，表示各种名字的字符串
3. 常　数：无符号数、布尔常数、字符串常数等
4. 分界符：+、-、*、/、...

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

词法分析程序的输出——单词的内部形式

单词类别	单词值
整型	58
保留字	“for”

几种常用的单词内部形式：

1、按单词种类分类

2、保留字和分界符采用一符一类

3、标识符和常数的单词值又为指示字（指针值）

1、按单词种类分类

类别编码	单词值

单词名称	类别编码	单词值
标识符	1	内部字符串
无符号常数(整)	2	整数值
无符号浮点数	3	数值
布尔常数	4	0 或 1
字符串常数	5	内部字符串
保留字	6	保留字或内部编码
分界符	7	分界符或内部编码

2、保留字和分界符采用一符一类

单词名称	类别编码	单词值
标识符	1	内部字符串
无符号常数(整)	2	整数值
无符号浮点数	3	数值
布尔常数	4	0 或 1
字符串常数	5	内部字符串
BEGIN	6	-
END	7	-
FOR	8	-
DO	9	-
……	…….	…..
:	20	-
+	21	-
*	22	-
,	23	-
(	…….	--

```
while (y < z) {
    int x = a + b;
    y += x;
}
```

w	h	i	l	e		(	y	<	z	)		{	\n
\t	i	n	t		x	=	a	+	b	;	\n	\t	y
	+	=		x	;	\n	}						

```
T_While
T_LeftParen
T_Identifier y
T_Less
T_Identifier z
T_RightParen
T_OpenBrace
T_Int
T_Identifier x
T_Assign
T_Identifier a
T_Plus
T_Identifier b
T_Semicolon
T_Identifier y
T_PlusAssign
T_Identifier x
T_Semicolon
T_CloseBrace
```

保留字: while， int

标识符: x, y, z, a, b

分隔符: _（空格）, \n, \t

token, symbol

单词，符号

规则从哪里来？

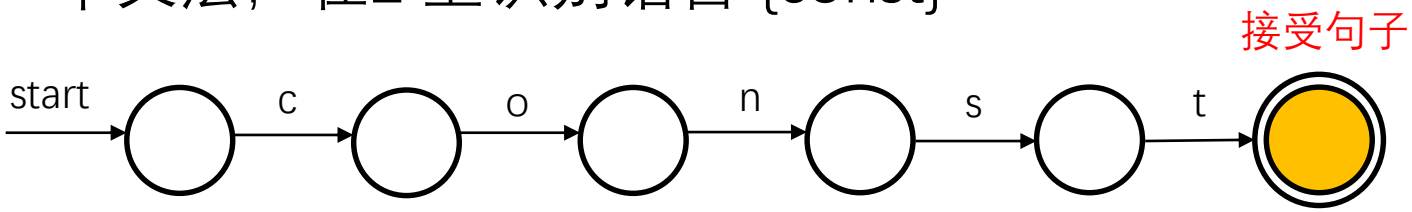
文法

Source: Stanford CS143 (2012)

识别一个“单词（token, symbol）”的文法：

- **保留字：**

构造一个文法，在Σ^*上识别语言 {const}



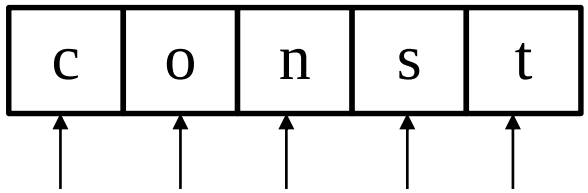
G(Z): Z::->’c’A

A::->’o’B

B::->’n’C

C::->’s’D

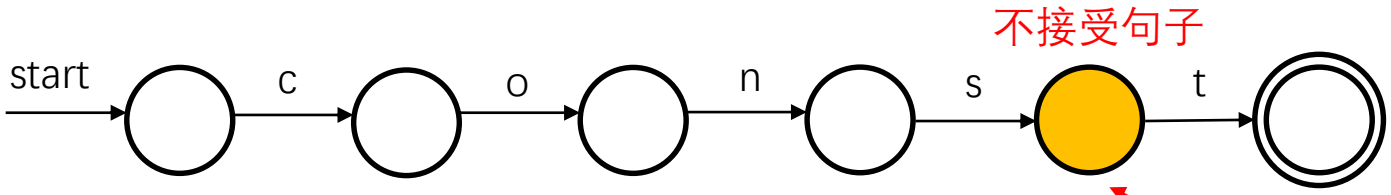
D::->’t’



识别一个“单词（token, symbol）”的文法：

- **保留字：**

构造一个文法，在Σ^*上识别语言 {const}



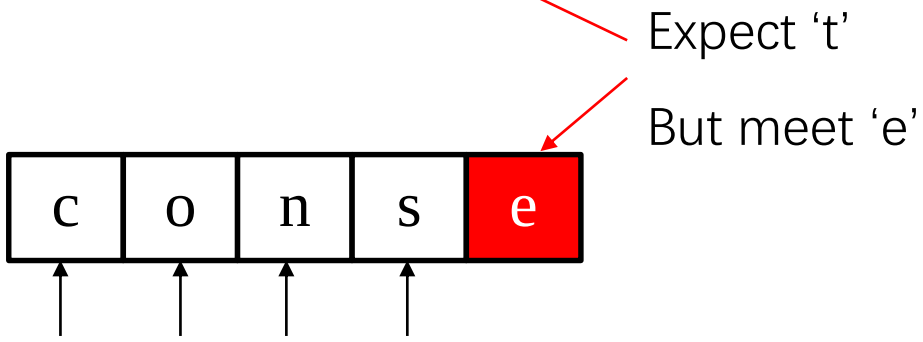
G(Z): Z::->’c’A

A::->’o’B

B::->’n’C

C::->’s’D

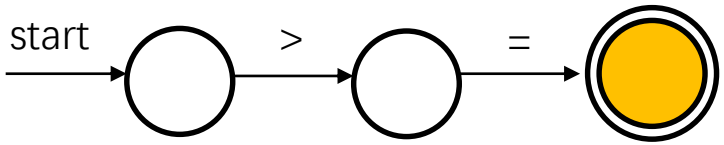
D::->’t’



识别一个“单词（token, symbol）”的文法：

♦ 分隔符：

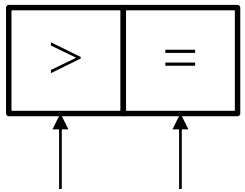
构造一个文法，在Σ^*上识别语言 {>=}



接受句子

G(Z): Z::->'>'A

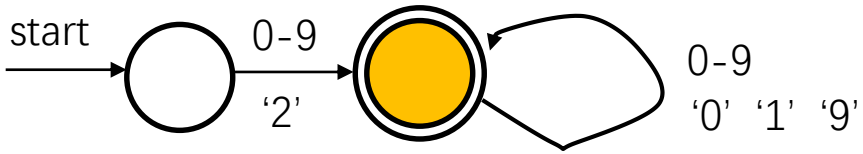
A::->'='



识别一个“单词（token, symbol）”的文法：

♦ 常量：（以PL0文法为例，p399，无符号整数）

构造一个文法，在Σ^*上识别语言 {const}

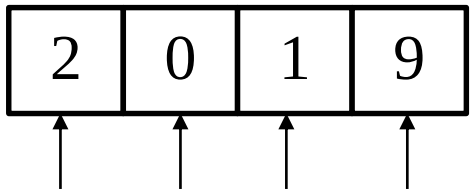


接受句子

G(<无>):

<无>::-><数字>{<数字>}

<数字>::->’0’|’1’|…|’8’|’9’

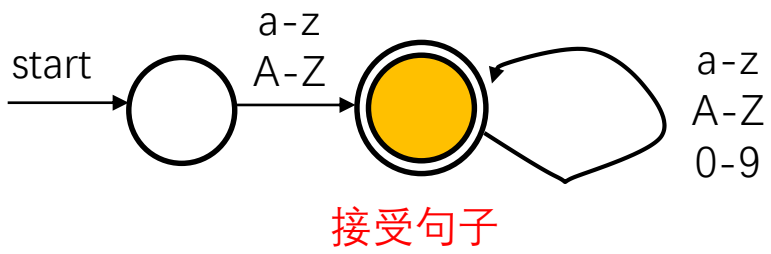


识别 ‘2019’

识别一个“单词（token, symbol）”的文法：

♦ 标识符：（以PL0文法为例，p399）

构造一个文法，在Σ^*上识别语言 {<无符号整数>}

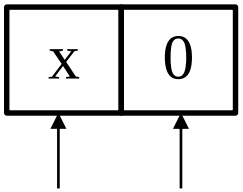


G(<标识符>):

<标识符>::-><字母>{<字母><数字>}

<数字>::->’0’|’1’|…|’8’|’9’

<字母>::->’a’|’b’|…|’z’|’A’|…|’Z’

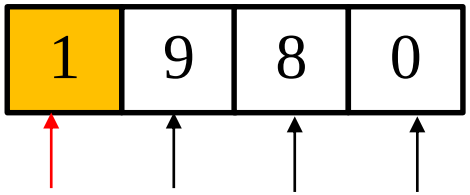


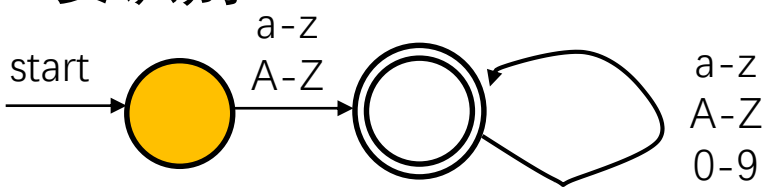
识别 ‘x0’

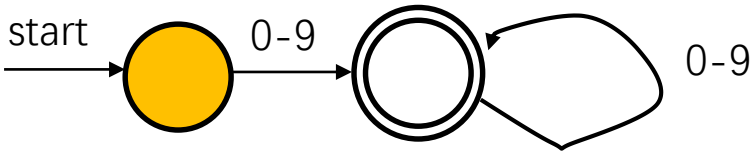
识别一个“单词（token, symbol）”的文法：

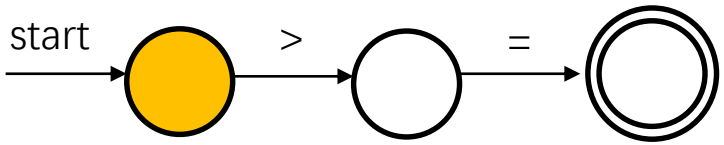
- 识别一个字符时，多个识别器可以同时工作

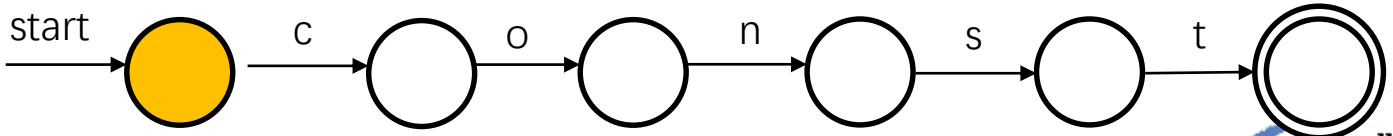
例如：要识别‘1980’:







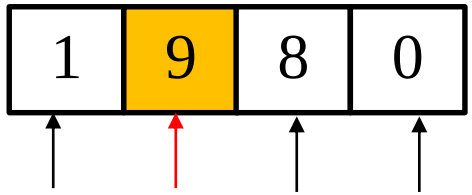


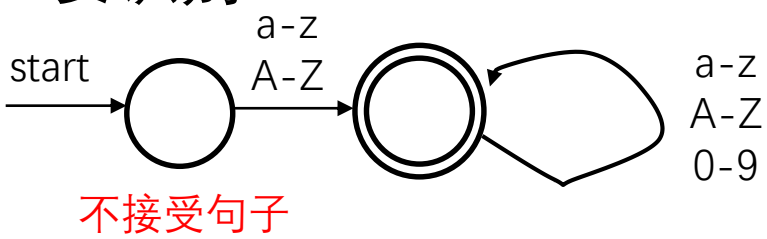


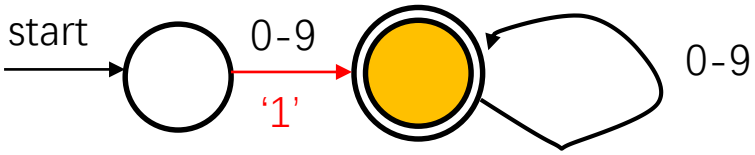
识别一个“单词（token, symbol）”的文法：

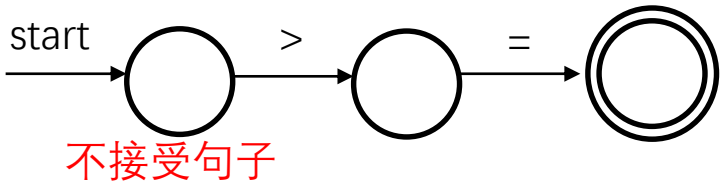
- 识别一个字符时，多个识别器可以同时工作

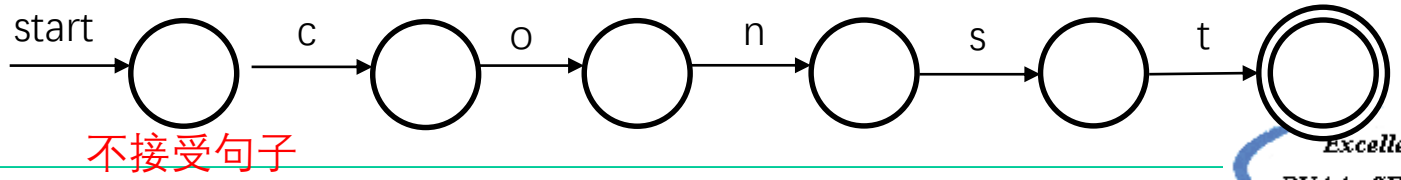
例如：要识别‘1980’:







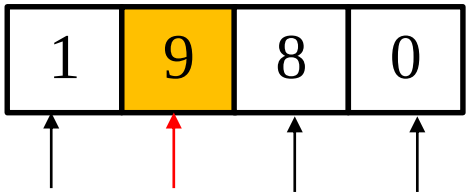


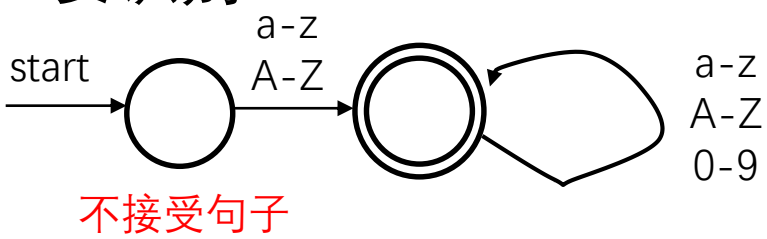


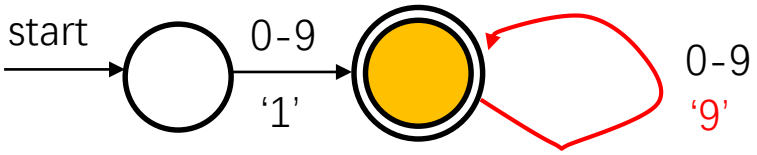
识别一个“单词（token, symbol）”的文法：

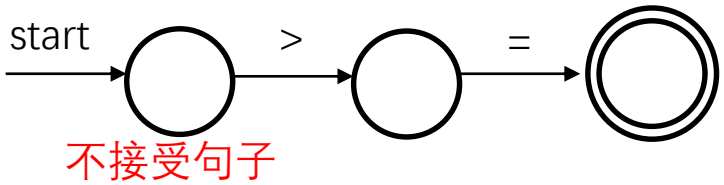
- 识别一个字符时，多个识别器可以同时工作

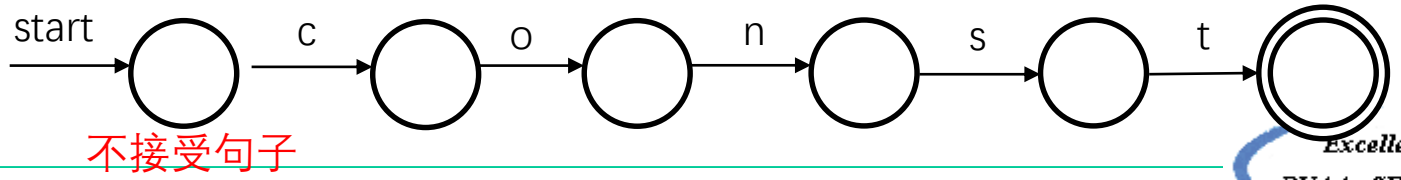
例如：要识别‘1980’:







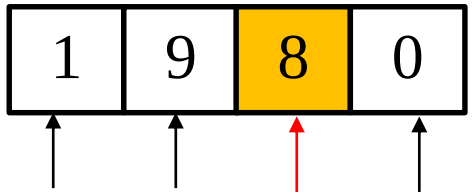


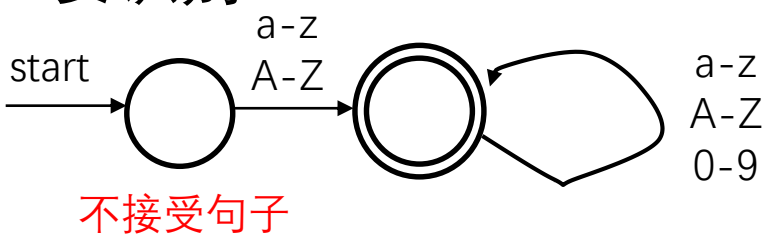


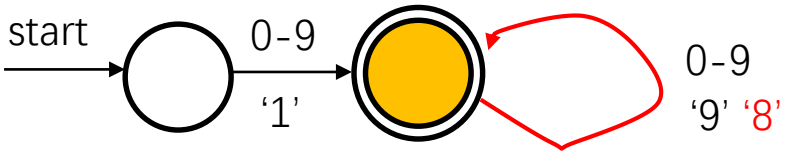
识别一个“单词（token, symbol）”的文法：

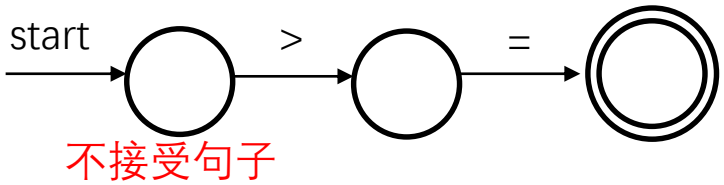
- 识别一个字符时，多个识别器可以同时工作

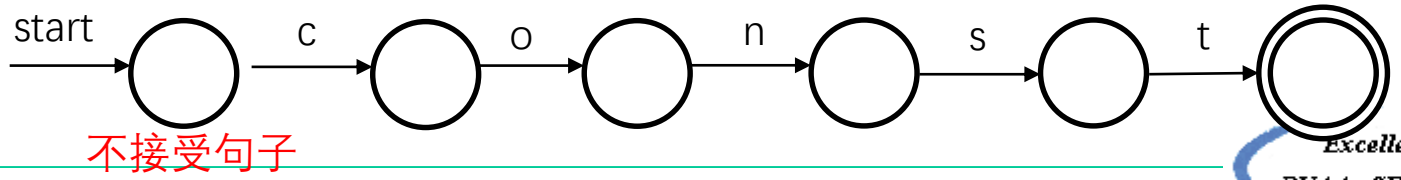
例如：要识别‘1980’:







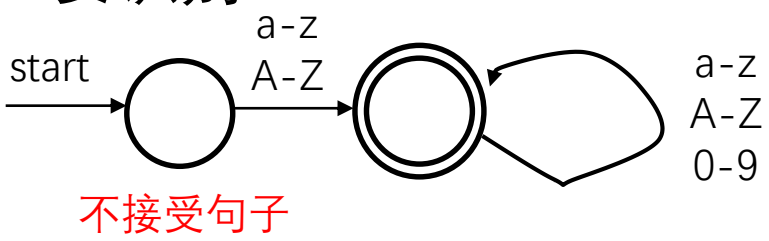




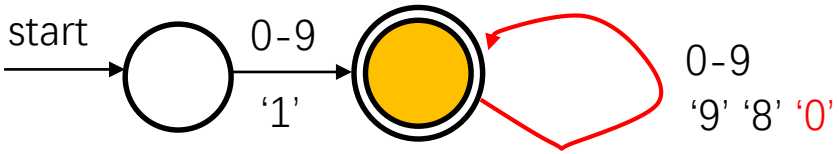
识别一个“单词（token, symbol）”的文法：

- 识别一个字符时，多个识别器可以同时工作

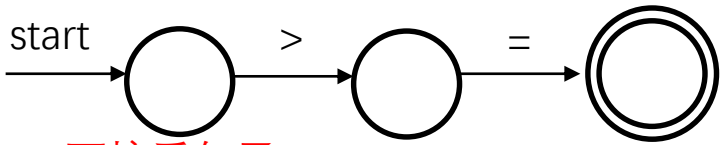
例如：要识别‘1980’:



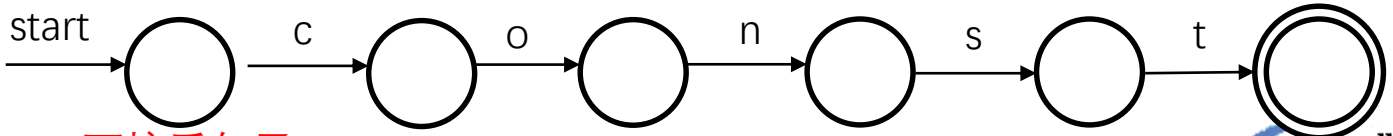
不接受句子



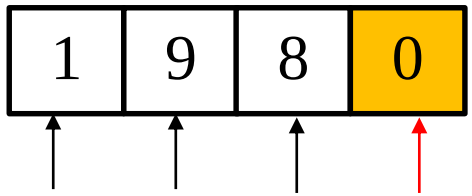
接受句子



不接受句子



不接受句子



识别结果：token

- 类别=2 （无符号常数）
- 值=1980

文法和状态图

识别一个“单词（token, symbol）”的文法：

- 按照乔姆斯基的文法体系，这些文法属于?

G(Z): Z::->’c’A
A::->’o’B
B::->’n’C
C::->’s’D
D::->’t’

G(Z): Z::->’>’A
A::->’=‘

G(<无>):
<无>::-><数字>{<数字>}
<数字>::->’0’|’1’|…|’8’|’9’

G(<标识符>):
<标识符>::-><字母>{<字母><数字>}
<数字>::->’0’|’1’|…|’8’|’9’
<字母>::->’a’|’b’|…|’z’|’A’|…|’Z’

识别一个“单词（token, symbol）”的文法：

- **按照乔姆斯基的文法体系，这些文法属于?**

3型文法、线性文法、正则文法

G(Z): Z::->’c’A
A::->’o’B
B::->’n’C
C::->’s’D
D::->’t’

G(Z): Z::->’>’A
A::->’=‘

G(<无>):
<无>::-><数字>{<数字>}
<数字>::->’0’|’1’|…|’8’|’9’

G(<标识符>):
<标识符>::-><字母>{<字母><数字>}
<数字>::->’0’|’1’|…|’8’|’9’
<字母>::->’a’|’b’|…|’z’|’A’|…|’Z’

3.3 正则文法和状态图

- **状态图的画法（根据文法画出状态图）**

例如：正则文法

Z::= U0 |V1

U ::=Z1 |1

V ::=Z0 | 0

左线性文法。该文法所定义的语言为：

$L(G[Z]) = \{ B^n \mid n>0 \}$，其中 B= {01,10}

例：正则文法

Z::= U0 |V1

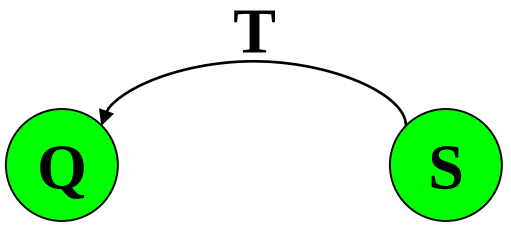
U ::=Z1 |1

V ::=Z0 | 0

左线性文法的状态图的画法：

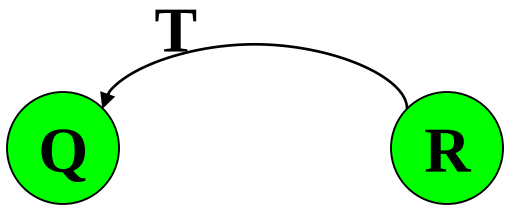
1. **令G的每个非终结符都是一个状态；**
2. **设一个开始状态S；**
3. **若Q::=T，Q ∈Vn,T ∈Vt, 则：**

   V::->0



4. **若Q::=RT, Q、R∈Vn,T ∈Vt, 则：**

   U::-> Z1



5. **按自动机方法，可加上开始状态和终止状态标志。**

1. **每个非终结符设一个状态；**
2. 设一个开始状态S；
3. 若Q::=T，Q ∈Vn,T ∈Vt，
4. 若Q::=RT, Q、R∈Vn,T ∈Vt，
5. 加上开始状态和终止状态标志

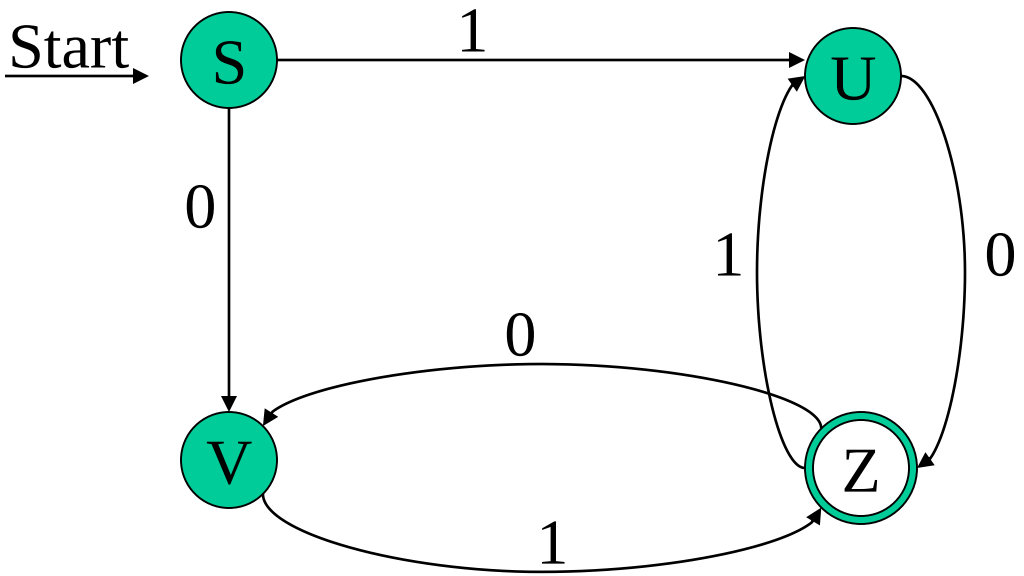
例如：正则文法

Z::= U0 |V1

U ::=Z1 |1

V ::=Z0 | 0

其状态图为：

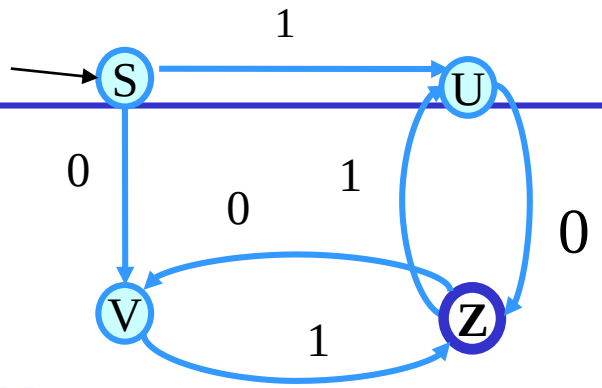


根据状态图：

Z::= U0 |V1

U ::=Z1 |1

V ::=Z0 | 0



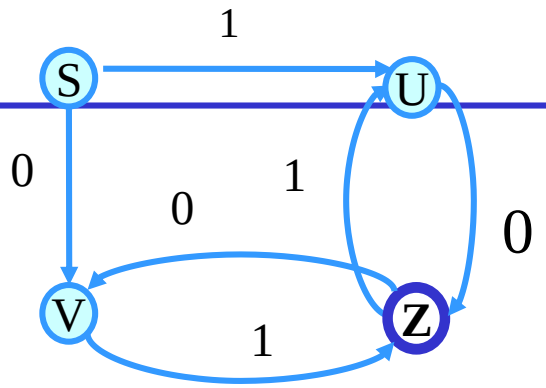
• 识别算法

利用状态图可按如下步骤分析和识别字符串x：

1、置初始状态为当前状态，从x的最左字符开始，重复步骤2，直到x右端为止。

2、扫描x的下一个字符，在当前状态所射出的弧中找出标记有该字符的弧，并沿此弧过渡到下一个状态；如果找不到标有该字符的弧，那么x不是句子，过程到此结束；如果扫描的是x的最右端字符，并从当前状态出发沿着标有该字符的弧过渡到下一个状态为终止状态Z，则x是句子。

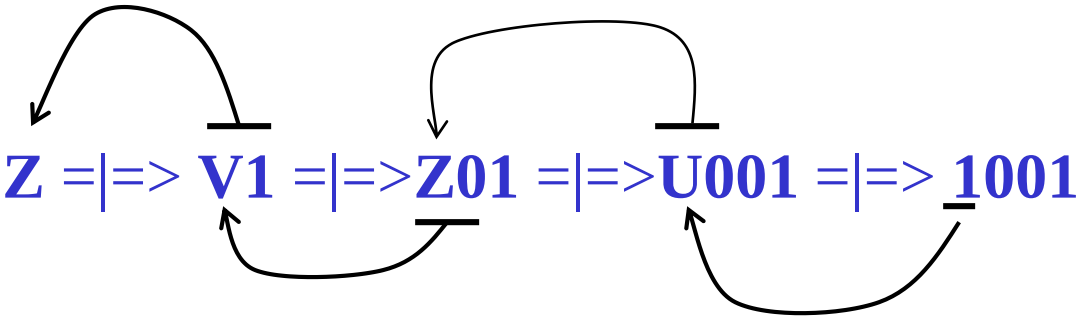
例：x=0110 和1011

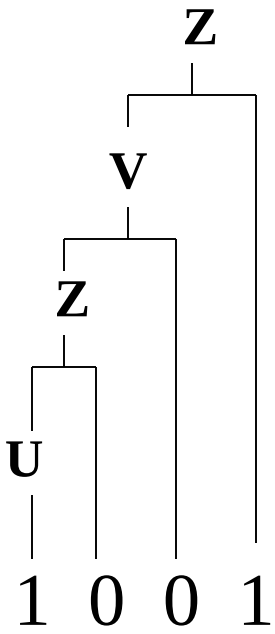


- **问题：**

1、上述分析过程是属于自底向上分析？还是自顶向下分析？

2、怎样确定句柄？





词法分析

因此：给定左线性文法，我们可以得到状态图

左右线性文法等价：从左（右）线性文法，可以构造等价的右（左）线性文法

G(Z): Z::->'c'A
A::->'o'B
B::->'n'C
C::->'s'D
D::->'t'

G(Z): Z::->A't'
A::->B's'
B::->C'n'
C::->D'o'
D::->'c'

```
while (y < z) {
    int x = a + b;
    y += x;
}
```

w	h	i	l	e		(	y	<	z	)		{	\n
\t	i	n	t		x	=	a	+	b	;	\n	\t	y
	+	=		x	;	\n	}						

```
T_While
T_LeftParen
T_Identifier y
T_Less
T_Identifier z
T_RightParen
T_OpenBrace
T_Int
T_Identifier x
T_Assign
T_Identifier a
T_Plus
T_Identifier b
T_Semicolon
T_Identifier y
T_PlusAssign
T_Identifier x
T_Semicolon
T_CloseBrace
```

保留字: while， int

标识符: x, y, z, a, b

分隔符: _（空格）, \n, \t

token, symbol

单词，符号

规则从哪里来？

（左、右）线性文法

状态图的构造

Source: Stanford CS143 (2012)

回顾：文法的其它表示法

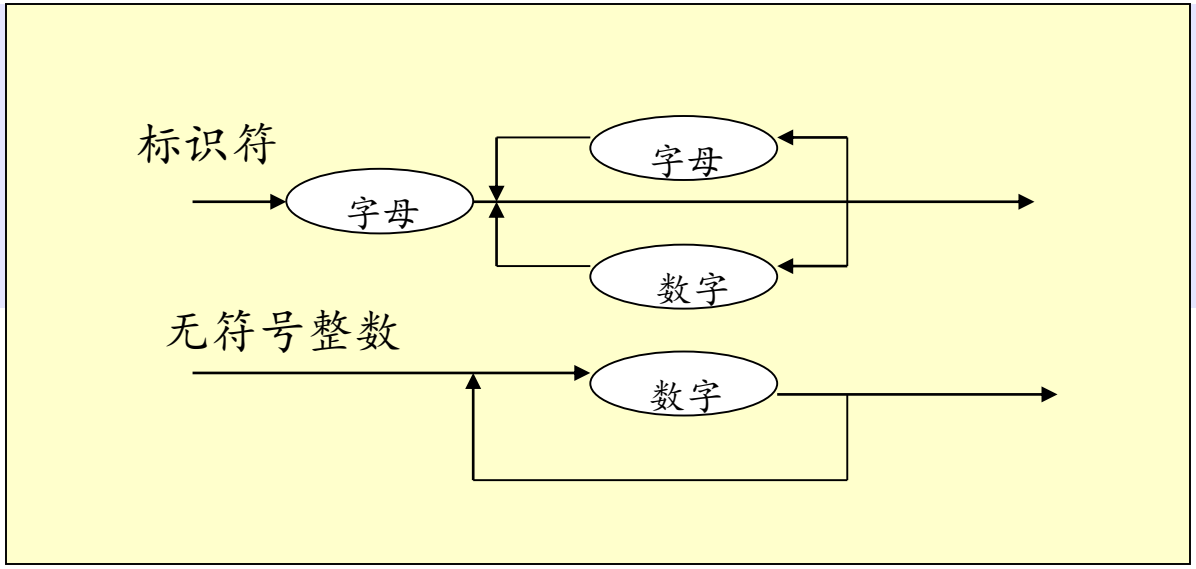
<标识符>::=字母{字母|数字}

<无符号整数>::=数字{数字}

1、扩充的BNF表示

- BNF的元符号： <, >, ::=, |
- 扩充的BNF的元符号： <, >, ::=, |, {, }, [,], (,)

2、语法图



3.4 词法分析程序的设计与实现

词法规则 → 状态图 → 词法分析程序

3.4.1 文法及其状态图

语言的单词符号

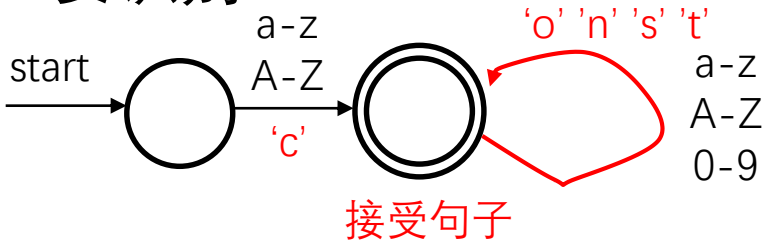
- 标识符
- 保留字(标识符的子集)
- 无符号整数
- 单分界符 + * : , ( )
- 双分界符 :=

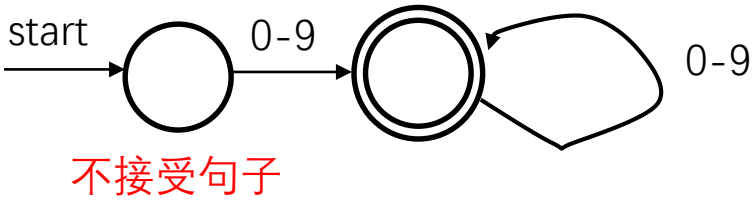
说明：1. 空格的作用　　2. 实数的表示

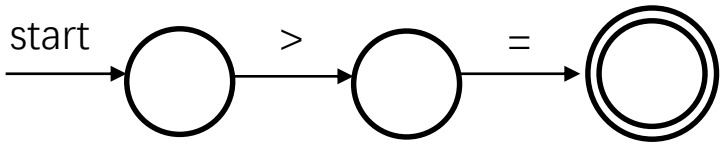
识别一个“单词（token, symbol）”的文法：

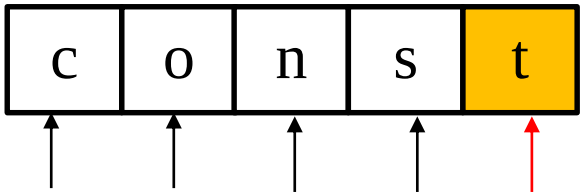
- 复习一个例子：识别一个字符时，多个识别器可以同时工作

例如： 要识别‘const’:









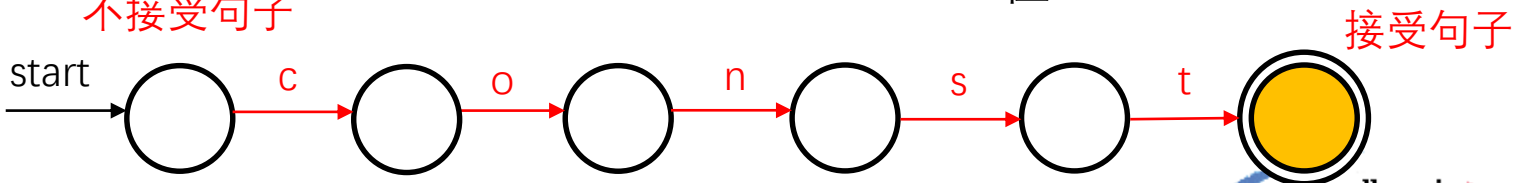
识别结果：token

- 类别=1 （标识符）
- 值=‘const’

识别结果：token

- 类别=6 （保留字）
- 值=const

?



完成词法分析的挑战

- **词法更加有效的表示方法?**

对于语言 {if}, {while}, 其实例很清晰（一对一）

对于“标识符”、“整数常量”，相对复杂 (采用状态图)

- **同一个字符串有多个词法规则可用时，如何选择?**

例如：从文法上，while 可以识别为标识符，也可以识别为保留字，如何选择

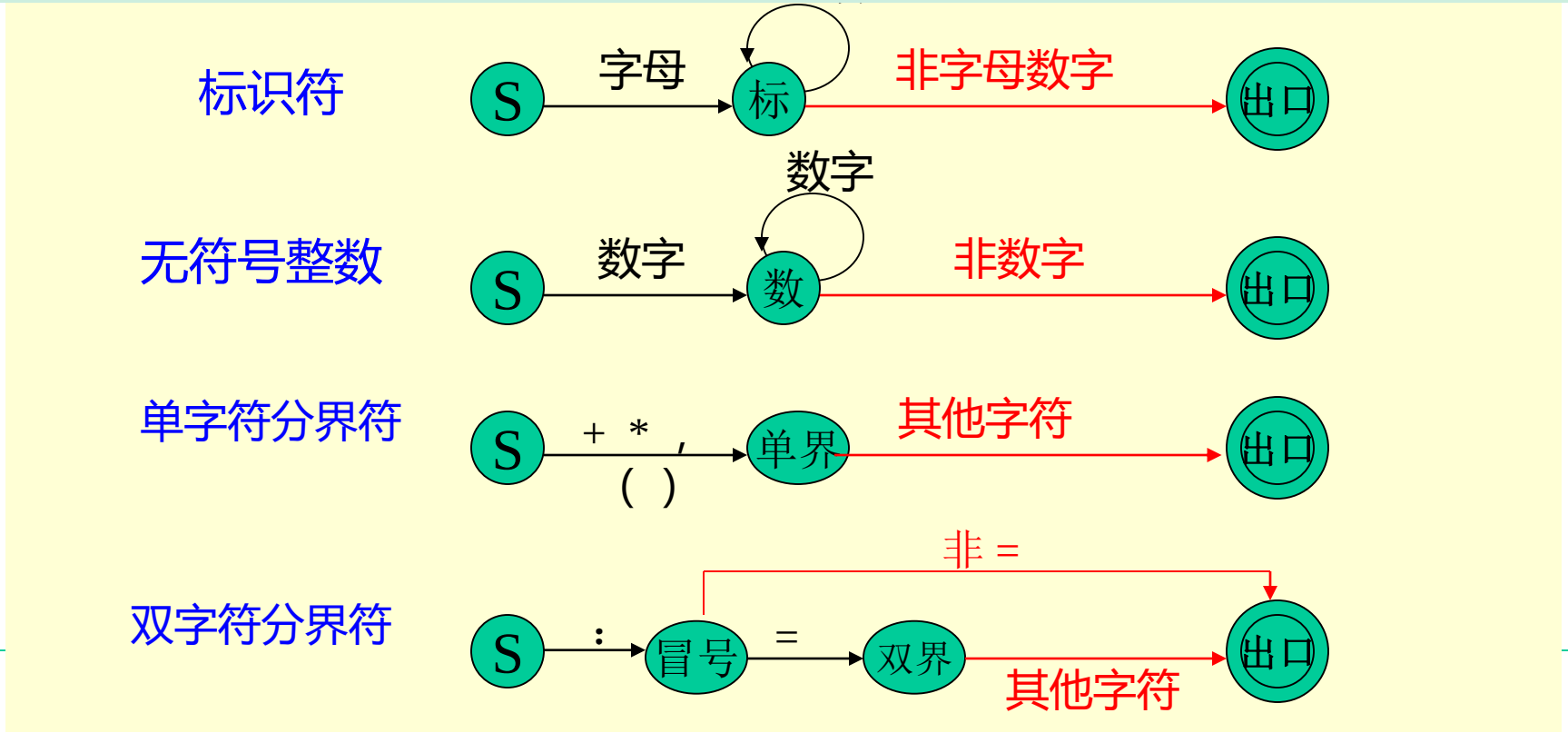
(长度匹配优先！例如：识别 ‘>=‘ 识别到一半时可匹配 ‘>’)

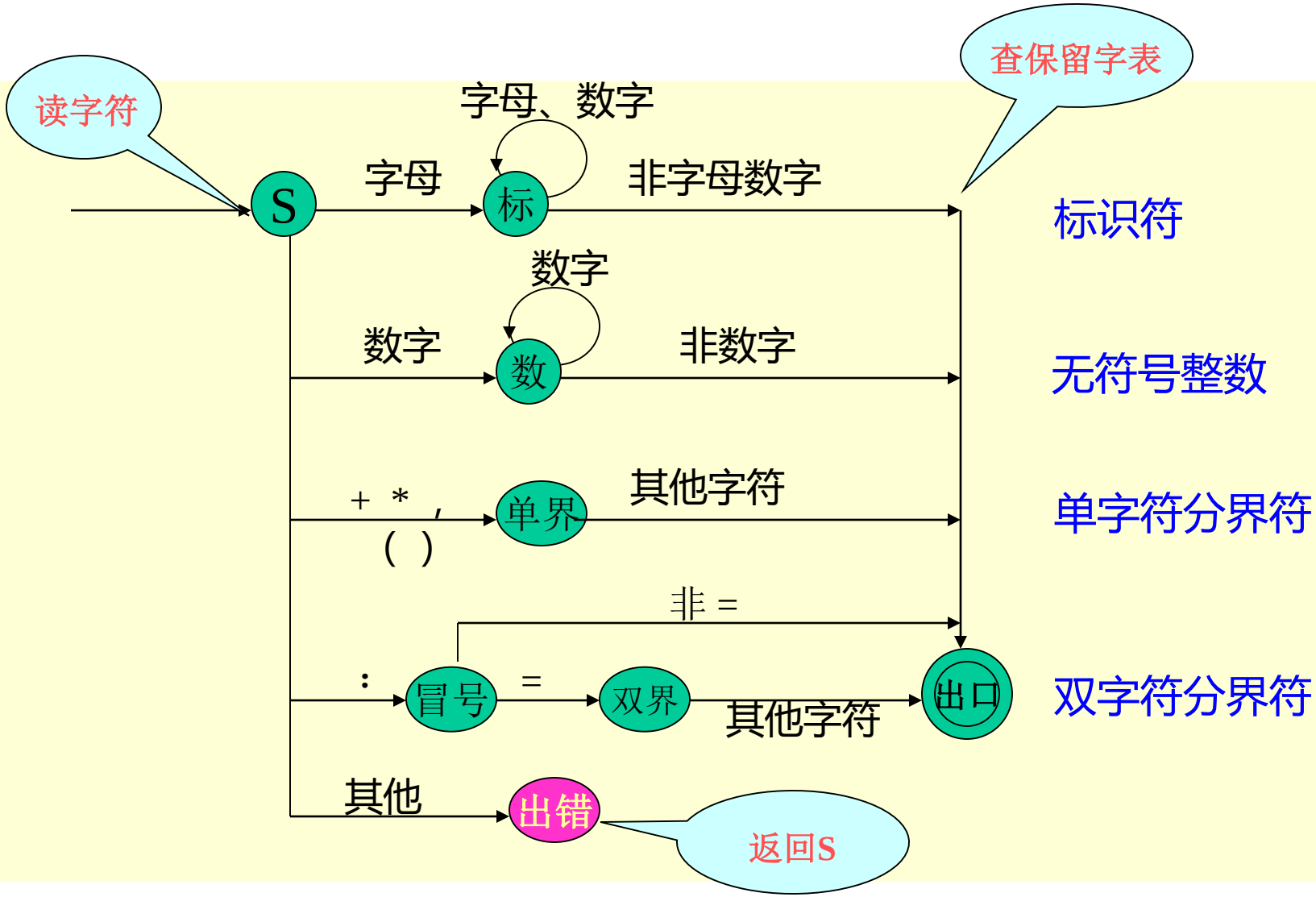
(人为增加优先级！例如，定义“保留字”优先)

技巧：通过“偷看”一个字符决定采用哪条规则

- 文法：**1. <标识符>::= 字母 | <标识符>字母 | <标识符>数字**
  2. <无符号整数>::=数字 | <无符号整数>数字
  3. <单字符分界符>::= : | + | * | , | (|)
  4. <双字符分界符>::= <冒号>=
  5. <冒号>::= :

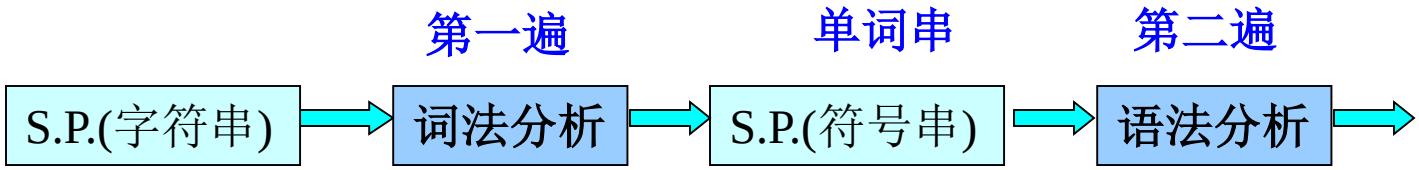
注意：这里不是严格的“状态图”，“出口”也不是“终止态”




读字符
查保留字表
字母、数字
字母
非字母数字
S
标
标识符
数字
数字
非数字
数
无符号整数
+ * ,
( )
其他字符
单界
单字符分界符
非 =
:
冒号
=
双界
出口
其他字符
双字符分界符
其他
出错
返回S

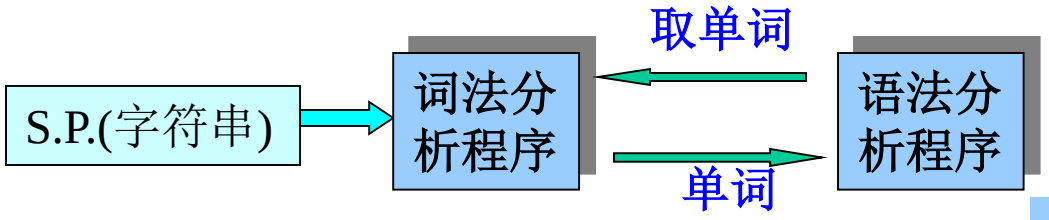
实现方案：基本上有两种

1.词法分析单独作为一遍



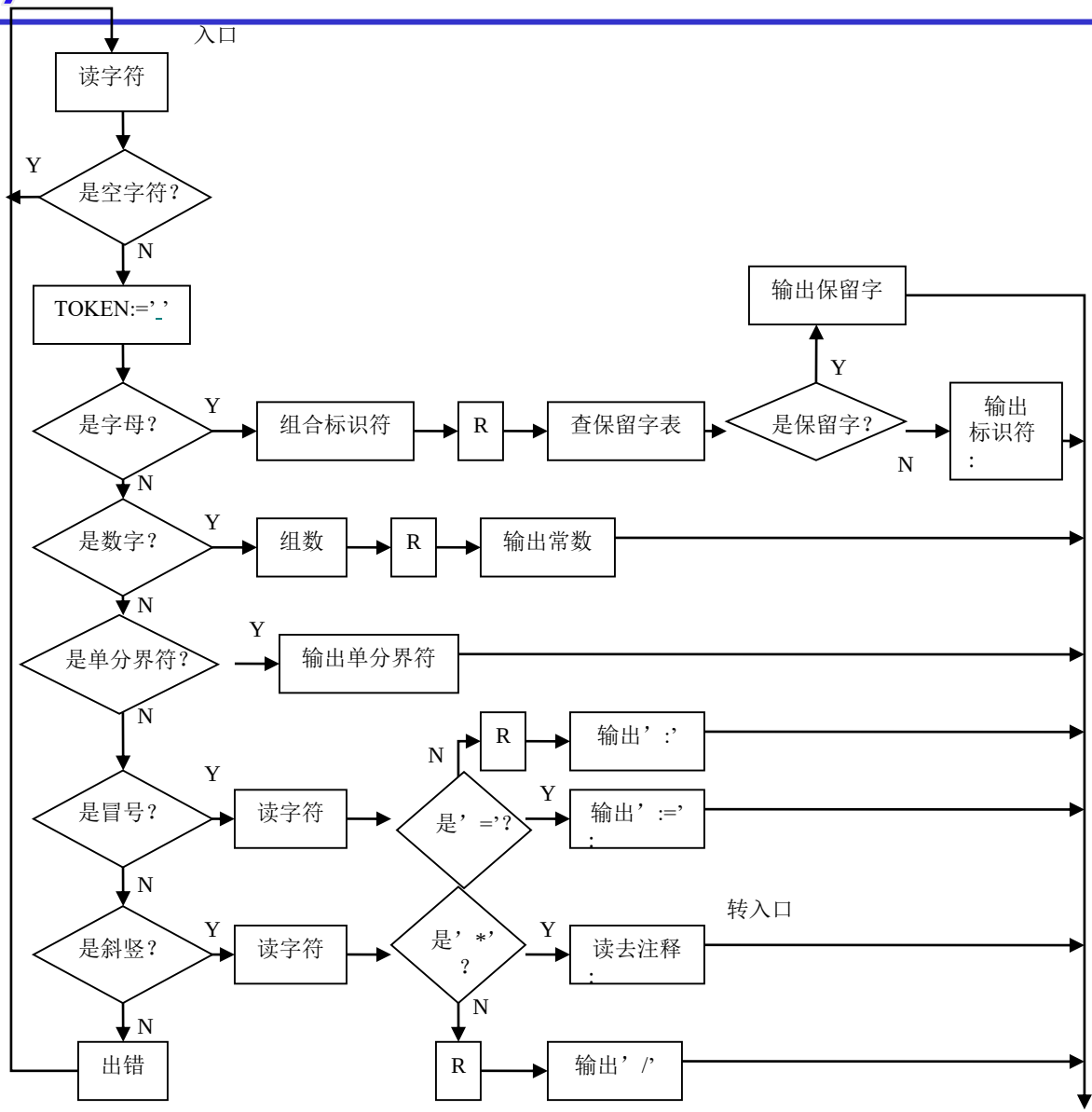
优点: 结构清晰、
各遍功能单一
缺点：效率低

2.词法分析程序作为单独的子程序



优点: 效率高

见教材


入口
读字符
是空字符？
Y
N
TOKEN:=’ ’
是字母？
Y
组合标识符
R
查保留字表
是保留字？
Y
N
输出保留字
输出标识符
：
是数字？
Y
组数
R
输出常数
是单分界符？
Y
输出单分界符
N
是冒号？
Y
读字符
是’=’?
N
R
输出’:’
Y
输出’:=’
：
是斜竖？
Y
读字符
是’*’？
Y
读去注释
：
转入口
N
出错
R
输出’/’
出口

3.4.2 状态图的实现——构造词法分析程序

1. 单词及内部表示

2. 词法分析程序需要引用的公共（全局）变量和过程

3. 词法分析程序算法

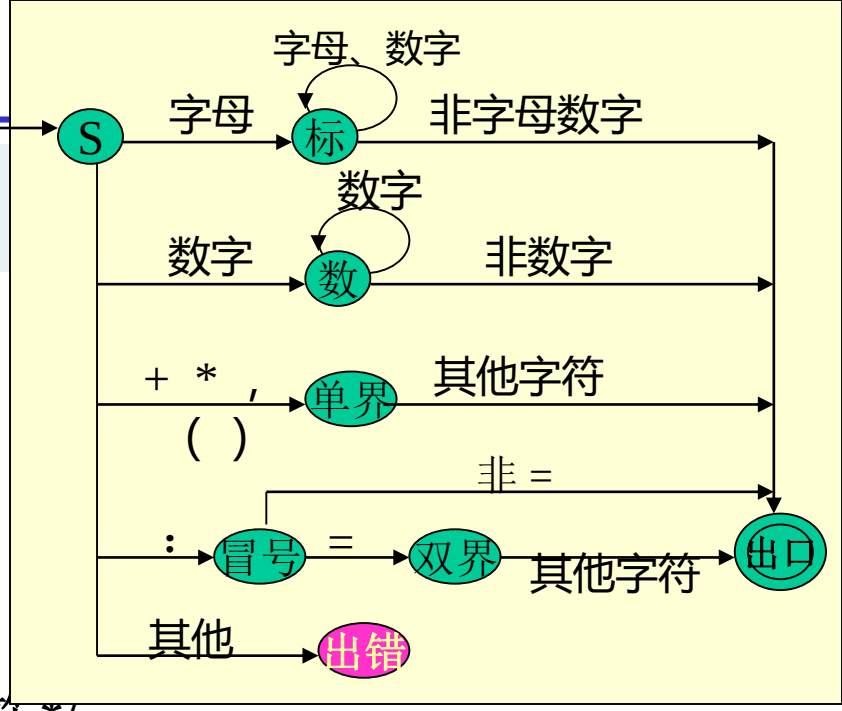
1.单词及内部表示:　保留字和分界符采用一符一类

单词名称	类别编码	记忆符	单词值
BEGIN	1	BeginSym	-
END	2	EndSym	-
FOR	3	ForSym	-
DO	4	DoSym	-
IF	5	IfSym	-
THEN	6	ThenSym	-
ELSE	7	ElseSym	-
标识符	8	IdSym	内部字符串
常数(整)	9	IntSym	整数值
:	10	ColonSym	-
+	11	PlusSym	-
*	12	StarSym	-
,	13	ComSym	-
(	14	LparSym	-
)	15	RparSym	-
:=	16	AssignSym	-

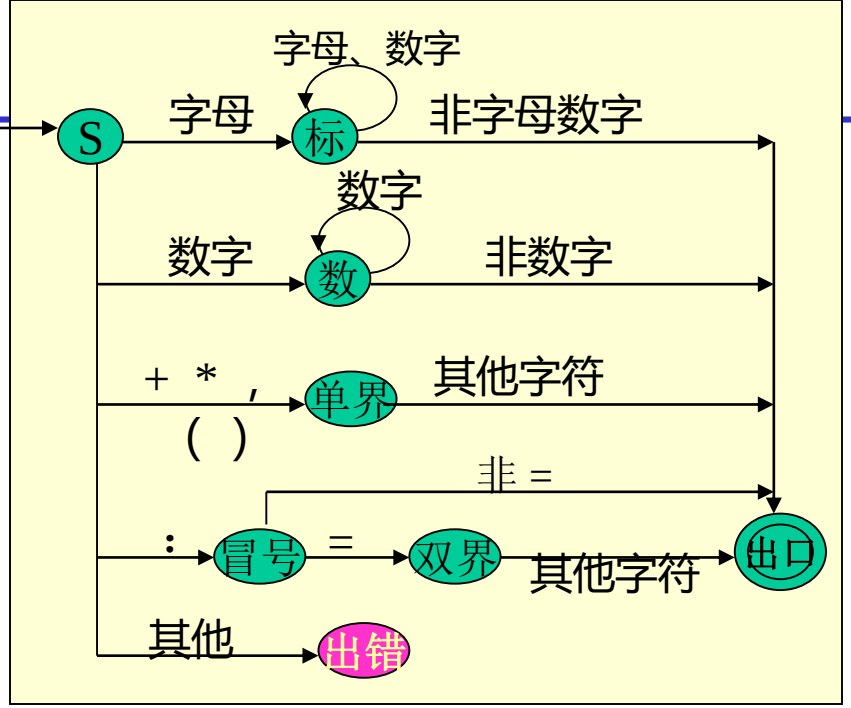
2.词法分析程序需要引用的公共（全局）变量和过程

名称	类别	功能
char	字符变量	存放当前读入的字符
token	字符数组	存放单词字符串
getchar	读字符过程	读字符到char,移动指针
getNBC	过程	反复调用getchar, 直至char进入一个非空白字符
CAT	过程	char与token连接
IsLetter 和 IsDigit	布尔函数	判断
UnGetCH	过程	读字符指针后退一个字符
RESERVE	布尔函数	判断token中的字符串 是保留字, 还是标识符
ATOI	函数	字符串到数字的转换
Error	过程	出错处理

3、词法分析程序算法




```
START:  token := ‘ ‘;     /*置token为空串*/
        getchar;  GetNBC;
  CASE  char OF
 ‘a’..’z’: BEGIN
             WHILE  IsLetter OR IsDigit DO
                    BEGIN CAT; getchar  END;
             UnGetCH;
             C:= RESERVE;     /* 返回0，为标识符 */
             IF C=0 THEN RETURN(‘IDSY’: TOKEN)
             ELSE RETURN (C,-)      /*C为保留字编码*/
           END;
 ‘0’..’9’:  BEGIN
             WHILE   IsDigit   DO
                    BEGIN CAT; getchar  END;
              UnGetCH;
              RETURN (‘INTSY’,ATOI)
           END;
‘+’ :   RETURN(‘PLUSSY’,-) ;
```




```
‘*’ :   RETURN(‘StarSym’,-) ;
‘,’ :   RETURN(‘ComSym’,-) ;
‘(’ :   RETURN(‘LparSym’,-) ;
‘)’ :   RETURN(‘RparSym’,-) ;
‘:’ :   BEGIN
            getchar;
            if  CHAR=‘=‘ THEN  RETURN(‘AssignSym’,-) ;
             UnGetCH;
            RETURN(‘ColonSym’,-) ;
        END
END OF CASE;
ERROR;
GOTO START;
```

练习：用C语言实现上述算法。

见教材

```
int getsym()        /*返回类别编码*、
{
        clearToken();
        while(isSpace()||isNewline()||isTab()) getchar();  /*读取字符，跳过空格、换行和Tab*/
    if(isLetter())                          /*判断当前字符是否是一个字母*/
     {
        while(isLetter()||isDigit())              /*将字符拼接成字符串*/
          { catToken();  getchar();}
        retract();                          /*指针后退一个字符*/
        int resultValue = reserver();             /*resultValue是查找保留字的返回值*/
        if(resultValue==0)  symbol= IDSY;       /*resultValue=0，token中的字符串为标识符*/
              else   symbol= resultValue;    /*否则token中的字符串为保留字*/
       }
    else if(isDigit())                      /*判断当前字符是否是一个数字*/
    {
        while(isDigit())                        /*将字符拼接成整数*/
          { catToken();  getchar();};
        retract();
        num= transNum(token);                   /*将token中的字符串转换成整数*/
        symbol= INTSY;                                  /*此时识别的单词是整数*/
     }
```

课程设计作业：词法分析

目标与要求：

- 根据具体文法编写词法分析程序，考核学生对词法分析方法的掌握情况，培养学生编写符合词法规则的词法分析程序的能力。
- 学生在理论课所学词法分析方法的基础上，设计实现词法分析程序，并按规定的格式输出单词信息
- 作业提交至教学平台，用5个测试程序进行测试，并进行相似性检查。根据输出结果与预期结果一致部分所占的比例给分。

目标与要求：

- 请根据给定的文法设计并实现词法分析程序，从源程序中识别出单词，记录其单词类别和单词值，输入输出及处理要求如下：
  1. 数据结构和与语法分析程序的接口请自行定义；类别码需按下表格式统一定义；
  2. 为了方便进行自动评测，约定：
     - 输入的被编译源文件统一命名为testfile.txt；
     - 输出的结果文件统一命名为output.txt；
     - 结果文件中每行按如下方式组织：
     - 单词类别码 单词的字符/字符串形式(中间仅用一个空格间隔)

目标与要求：

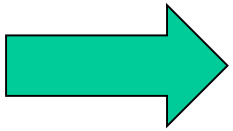
- 单词的类别码请统一按如下形式定义：

单词名称	类别码	单词名称	类别码	单词名称	类别码	单词名称	类别码
标识符	IDENFR	if	IFTK	-	MINU	=	ASSIGN
整型常量	INTCON	else	ELSETK	*	MULT	;	SEMICN
字符常量	CHARCON	do	DOTK	/	DIV	,	COMMA
字符串	STRCON	while	WHILETK	<	LSS	(	LPARENT
const	CONSTTK	for	FORTK	<=	LEQ	)	RPARENT
int	INTTK	scanf	SCANFTK	>	GRE	[	LBRACK
char	CHARTK	printf	PRINTFTK	>=	GEQ	]	RBRACK
void	VOIDTK	return	RETURNTK	==	EQL	{	LBRACE
main	MAINTK	+	PLUS	!=	NEQ	}	RBRACE

例如有如下程序段：

```
const int const1 = 1, const2
= -100;
const char const3 = '_';
int change1;
char change3;
int gets1(int var1,int var2){
    change1 = var1 + var2;
    return (change1);
}
```

则输出的结果文件具有的内容如右侧序列



```
CONSTTK const
INTTK int
IDENFR const1
ASSIGN =
INTCON 1
COMMA ,
IDENFR const2
ASSIGN =
MINU -
INTCON 100
SEMICN ;
CONSTTK const
CHARTK char
IDENFR const3
ASSIGN =
CHARCON _
SEMICN ;
INTTK int
IDENFR change1
SEMICN ;
CHARTK char
IDENFR change3
SEMICN ;
```

```
INTTK int
IDENFR gets1
LPARENT (
INTTK int
IDENFR var1
COMMA ,
INTTK int
IDENFR var2
RPARENT )
LBRACE {
IDENFR change1
ASSIGN =
IDENFR var1
PLUS +
IDENFR var2
SEMICN ;
RETURNTK return
LPARENT (
IDEN change1
RPARENT )
SEMICN ;
        RBRACE }
```

词法分析视频:

第三章作业:

P73 1,2,3(词法见65-67页)

谢谢！