



中国科学技术大学  
University of Science and Technology of China

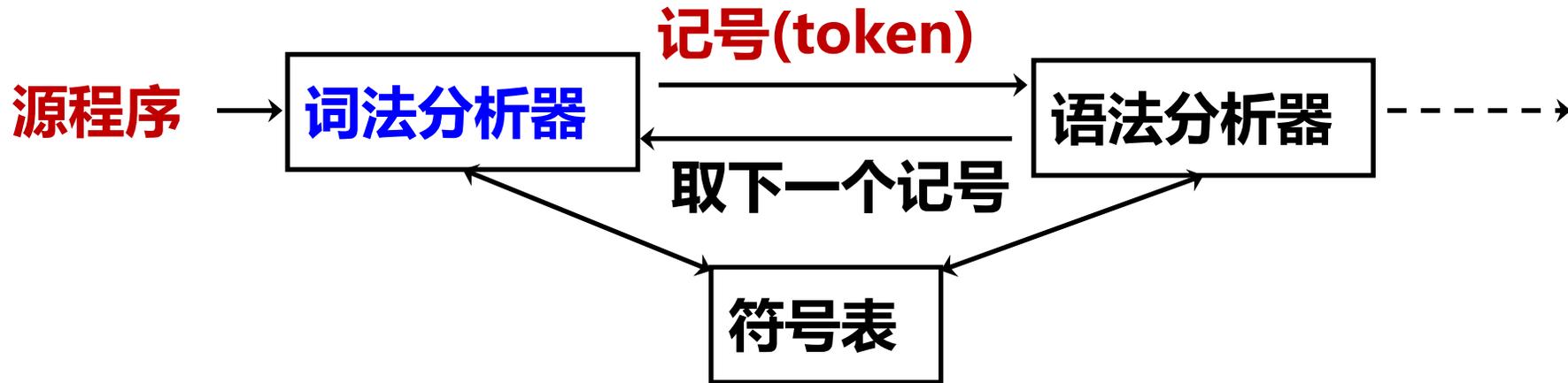
# 词法分析 I

《编译原理和技术(H)》、《编译原理(H)》

张昱

0551-63603804, [yuzhang@ustc.edu.cn](mailto:yuzhang@ustc.edu.cn)

中国科学技术大学  
计算机科学与技术学院



## □ 词法分析及要解决的问题

- 向前看(Lookahead)、歧义(Ambiguities)

## □ 词法分析器的自动生成

- 词法的描述：**正规式**；词法记号的识别：**转换图**
- **有限自动机：NFA、DFA**



## 2.1 词法记号及属性

- 词法单元(lexeme, 词素)
- 记号(token)
- 模式(pattern)



# 词法记号、词法单元、模式

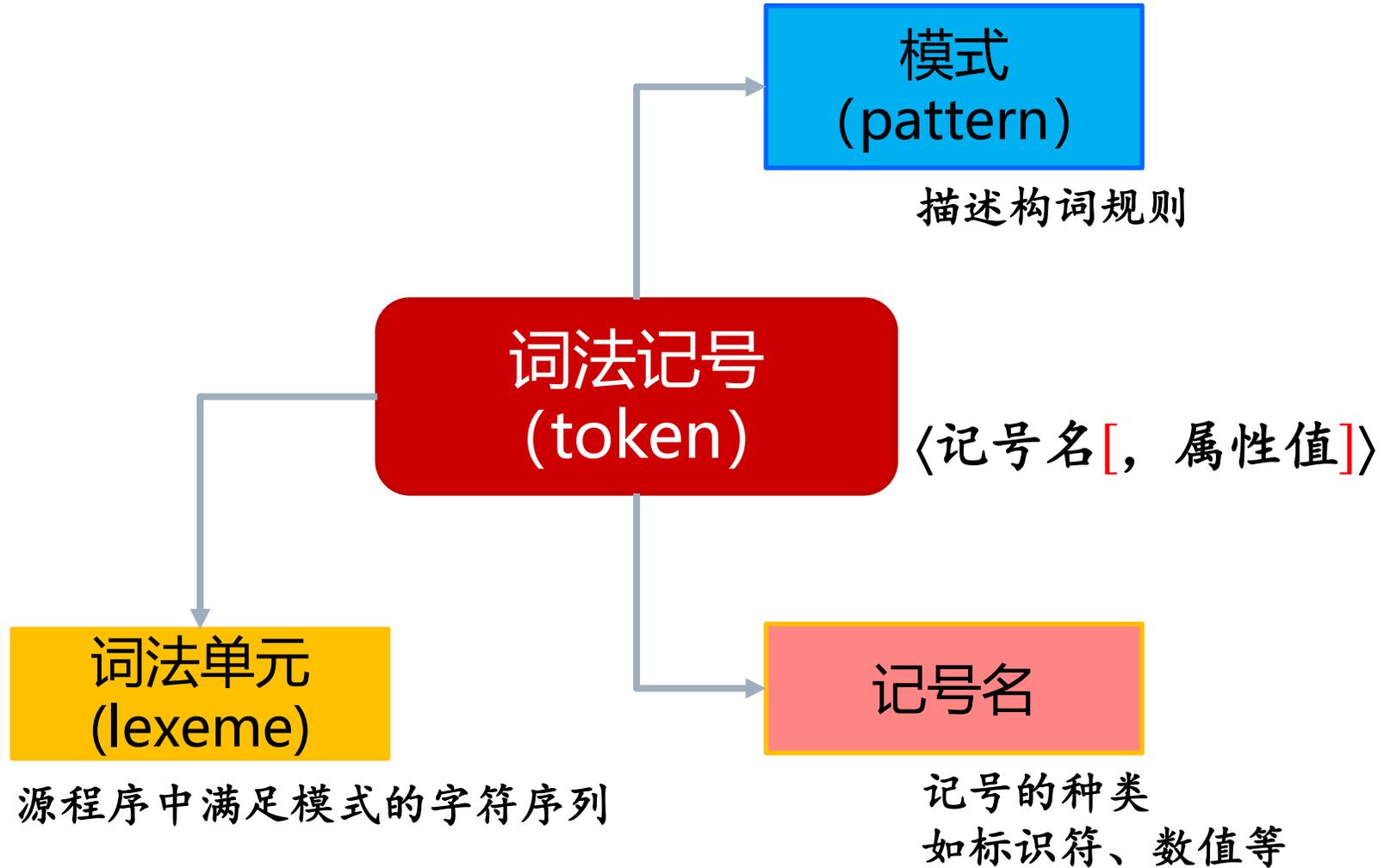
记号名	词法单元实例	模式的非形式描述
if	if	字符i, f
for	for	字符f, o, r
relation	<, <=, =, ...	< 或 <= 或 = 或 ...
id	sum, count, D5	由字母开头的字母数字串
number	3.1, 10, 2.8 E12	任何数值常数
literal	“seg. error”	引号“和”之间任意不含引号本身的字符串
whitespa	换行符	换行符\n

空白字：如空格、\t、换行  
无意义，被丢弃，不提供给语法分析器

- 1. 词法单元(lexeme):**  
又称 **单词** 或 **词素**  
源程序中具有某种词法含义的一个字符序列
- 2. 词法记号(token):**  
简称**记号**  
由记号名和可选的属性值组成
- 3. 模式(pattern)**  
描述属于该记号的词法单元的形式



# 词法记号、词法单元、模式





# 词法记号的属性

**position = initial + rate \* 60** 的记号

⟨记号名[, 属性值]⟩:

⟨**id**, 指向符号表中**position**条目的指针⟩

⟨**assign\_op**⟩

⟨**id**, 指向符号表中**initial**条目的指针⟩

⟨**add\_op**⟩

⟨**id**, 指向符号表中**rate**条目的指针⟩

⟨**mul\_op**⟩

⟨**number**, 整数值60⟩

符号表

1	<b>position</b>	...
2	<b>initial</b>	...
3	<b>rate</b>	...

**Lexeme**  
词法单元/单词/词素



# 词法定义中的问题

## □ 关键字 ≠ 保留字

- **关键字(keyword)**: 有专门的意义和用途, 如if、else
- **保留字**: 有专门的意义, 不能当作一般的标识符使用  
例如, C语言中的关键字是保留字

## □ 历史上词法定义中的一些问题

- 忽略空格带来的困难, 例如 Fortran

DO 8 I = 3.75 等同于 DO8I = 3.75

DO 8 I = 3,75

空格不是  
分隔符

- 关键字不保留

IF THEN THEN THEN=ELSE; ELSE ...



## 2.2 词法记号的描述与识别

- 描述：正规式
- 识别：转换图



## □ 术语

- **字母表**: 符号的有限集合, 例:  $\Sigma = \{0, 1\}$ 、ASCII、Unicode
- **串**: 符号的有穷序列, 例: 0110,  $\epsilon$
- **语言**: 字母表  $\Sigma$  上的一个串集  
 $\{\epsilon, 0, 00, 000, \dots\}$ ,  $\{\epsilon\}$ ,  $\emptyset$
- **句子**: 属于语言的串

词法单元是源程序中的字符序列 (字符串)  
字符串集合由称为**模式**的规则来描述

串  $s$  的长度是出现在  $s$  中符号的个数

## □ 串的运算

- **连接 (积)**  $xy$ ,  $s\epsilon = \epsilon s = s$
- **幂**  $s^0$  为  $\epsilon$ ,  $s^i$  为  $s^{i-1}s$  ( $i > 0$ )



**优先级:**  
幂 > 连接



## □ 语言的运算

语言表示字母表上的一个串集

■ 并:  $L \cup M = \{s \mid s \in L \text{ 或 } s \in M\}$

■ 连接:  $LM = \{st \mid s \in L \text{ 且 } t \in M\}$

■ 幂:  $L^0$ 是 $\{\epsilon\}$ ,  $L^i$ 是 $L^{i-1}L$

■ 闭包:  $L^* = L^0 \cup L^1 \cup L^2 \cup \dots$

■ 正闭包:  $L^+ = L^1 \cup L^2 \cup \dots$

优先级:

幂 > 连接 > 并

均是自左向右结合

## □ 例

$L: \{A, B, \dots, Z, a, b, \dots, z\}, D: \{0, 1, \dots, 9\}$

$L \cup D, LD, L^6, L^*, L(L \cup D)^*, D^+$



正规式(正则表达式)用来表示简单的语言, 叫做正规集

正规式	定义的语言	备注
$\varepsilon$	$\{\varepsilon\}$	
$a$	$\{a\}$	$a \in \Sigma$
$(r)$	$L(r)$	$r$ 是正规式
$(r)   (s)$	$L(r) \cup L(s)$	$r$ 和 $s$ 是正规式
$(r)(s)$	$L(r)L(s)$	$r$ 和 $s$ 是正规式
$(r)^*$	$(L(r))^*$	$r$ 是正规式

$((a)(b)^*) | (c)$ 可以写成 $ab^* | c$

优先级:  
闭包\* > 连接 > 选择 |



# 正规式遵守的一些代数定律

定律	描述
$\mathbf{r   s = s   r}$	是可交换的
$\mathbf{r   (s   t) = (r   s)   t}$	是可结合的
$\mathbf{(rs)t = r(st)}$	连接是可结合的
$\mathbf{r(s   t) = rs   rt; (s   t)r = sr   tr}$	连接对   是可分配的
$\mathbf{\varepsilon r = r\varepsilon = r}$	$\varepsilon$ 是连接的恒等元素
$\mathbf{r^* = (r   \varepsilon)^*}$	$\varepsilon$ 肯定出现在一个闭包中
$\mathbf{r^{**} = r^*}$	*是幂等的



□  $\Sigma = \{a, b\}$

■  $a \mid b$   $\{a, b\}$

■  $(a \mid b)(a \mid b)$   $\{aa, ab, ba, bb\}$

■  $aa \mid ab \mid ba \mid bb$   $\{aa, ab, ba, bb\}$

■  $a^*$  由字母 $a$ 构成的所有串的集合

■  $(a \mid b)^*$  由 $a$ 和 $b$ 构成的所有串的集合

□ 复杂的例子

$(00 \mid 11 \mid ((01 \mid 10)(00 \mid 11)^*(01 \mid 10)))^*$

句子: **01001101000010000010111001**



# 正规定义(regular definition)

- 对正规式命名, 使正规式表示简洁

$$d_1 \rightarrow r_1$$

$$d_2 \rightarrow r_2$$

...

$$d_n \rightarrow r_n$$

自底向上定义  
Bottom-up

- 各个 $d_i$  的名字都不同, 是新符号, 不在字母表  $\Sigma$  中
- 每个 $r_i$  都是  $\Sigma \cup \{d_1, d_2, \dots, d_{i-1}\}$  上的正规式
- C语言的标识符是字母、数字和下划线组成的串

$$\text{letter\_} \rightarrow A | B | \dots | Z | a | b / \dots | z / \_$$

$$\text{digit} \rightarrow 0 | 1 | \dots | 9$$

$$\text{id} \rightarrow \text{letter\_}(\text{letter\_} | \text{digit})^*$$



□ 无符号数集合，例1946, 11.28, 63E8, 1.99E-6

**digit**  $\rightarrow 0 \mid 1 \mid \dots \mid 9$

**digits**  $\rightarrow \text{digit digit}^*$

**optional\_fraction**  $\rightarrow \text{.digits} \mid \varepsilon$

**optional\_exponent**  $\rightarrow ( \text{E} ( + \mid - \mid \varepsilon ) \text{digits} ) \mid \varepsilon$

**number**  $\rightarrow \text{digits optional\_fraction optional\_exponent}$



## □ 无符号数集合，例1946, 11.28, 63E8, 1.99E-6

**digit**  $\rightarrow 0 \mid 1 \mid \dots \mid 9$

简记为 **[0-9]** --- 字符组

**digits**  $\rightarrow$  **digit digit\***

**optional\_fraction**  $\rightarrow$  **.digits**  $\mid \epsilon$

**optional\_exponent**  $\rightarrow$  **( E ( +  $\mid$  -  $\mid$   $\epsilon$  ) digits )  $\mid$   $\epsilon$**

**number**  $\rightarrow$  **digits optional\_fraction optional\_exponent**

## □ 简化的表示

**number**  $\rightarrow$  **digit<sup>+</sup> (.digit<sup>+</sup>)<sup>?</sup> (E(+|-) ? digit<sup>+</sup>)<sup>?</sup>**

注意区分? 和 \*

? 表示0个或1个, \*表示0个或多个, +表示1个或多个



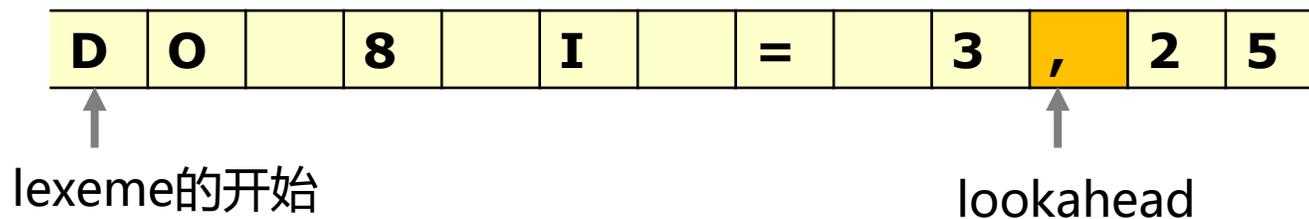
## 2.2 词法记号的描述与识别

- 描述：正规式
- 识别：转换图



## □ 词法分析

- 从左到右读取输入串，每次识别出一个token实例
- 可能需要“lookahead”来判断当前是否是token的结尾或下一个token的开始（尤其是对Fortran语言）





## □ 词法分析

- 从左到右读取输入串，每次识别出一个token实例
- 可能需要“lookahead”来判断当前是否是一个token的结尾、下一个token的开始（尤其是在Fortran语言中）
- 可能需要结合上下文来识别是否是关键字（当关键字不是保留字时）

```
if (then .gt. else) then  
    then = else  
else  
    else = then  
endif
```

需要结合上下文  
识别是否是关键字



# 词法分析器：实现

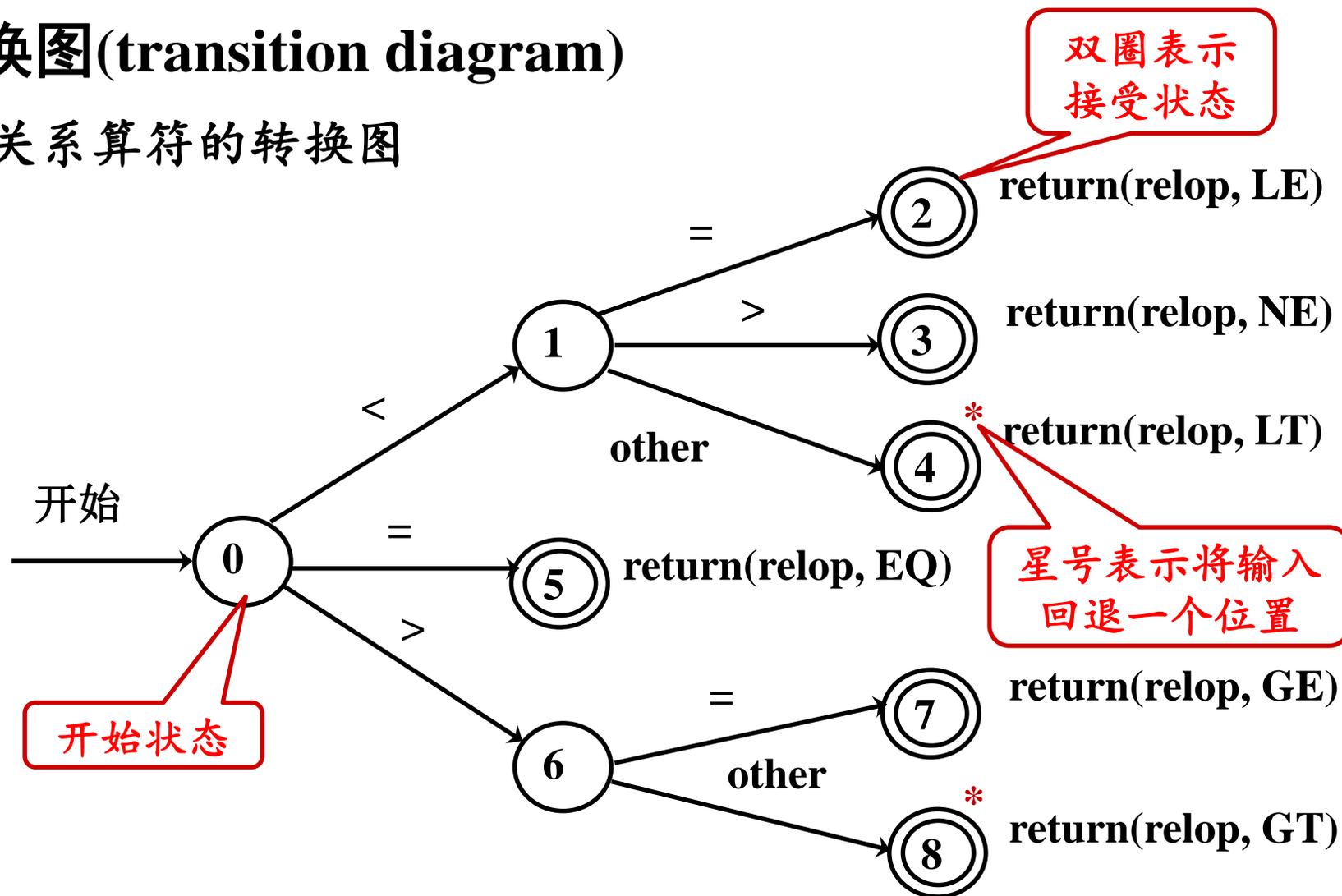
一个词法分析器的实现必须做两件事

1. 识别子串并对应到 tokens
2. 返回token的值或词法单元(lexeme, 词素)
  - 词法单元是子串 (token的实例)

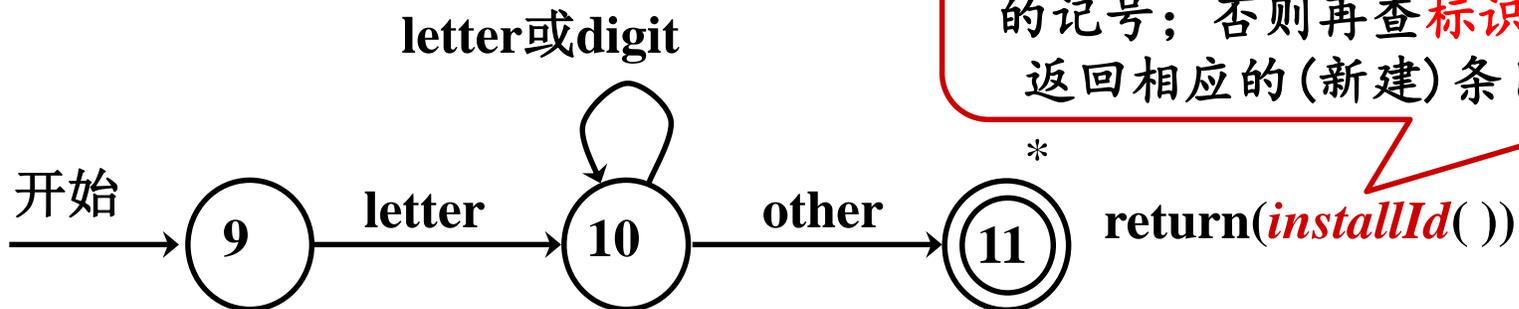


## □ 转换图(transition diagram)

### ■ 关系算符的转换图

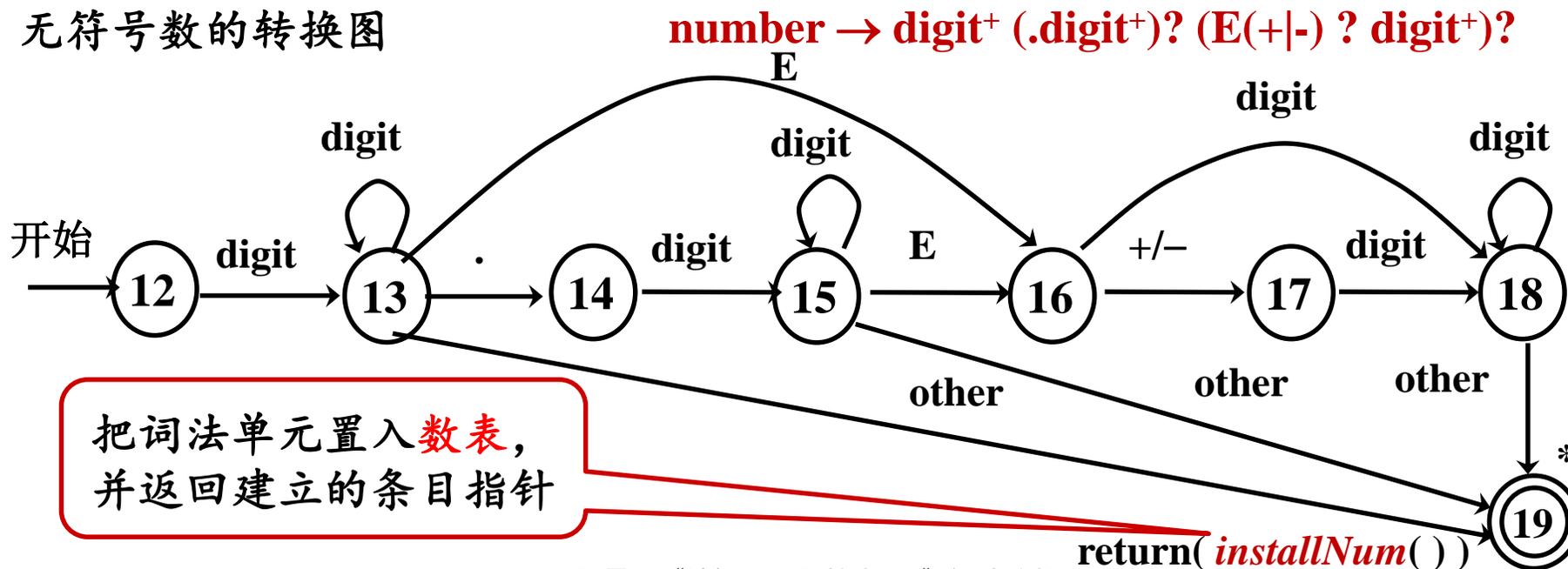


## 标识符和关键字的转换图



先查看**关键字表**，若当前词法单元构成关键字，则返回相应的记号；否则再查**标识符表**，返回相应的(新建)条目标针

## 无符号数的转换图



把词法单元置入**数表**，并返回建立的条目标针





# 基于转换图的词法分析

## 例：relop的转换图的概要实现

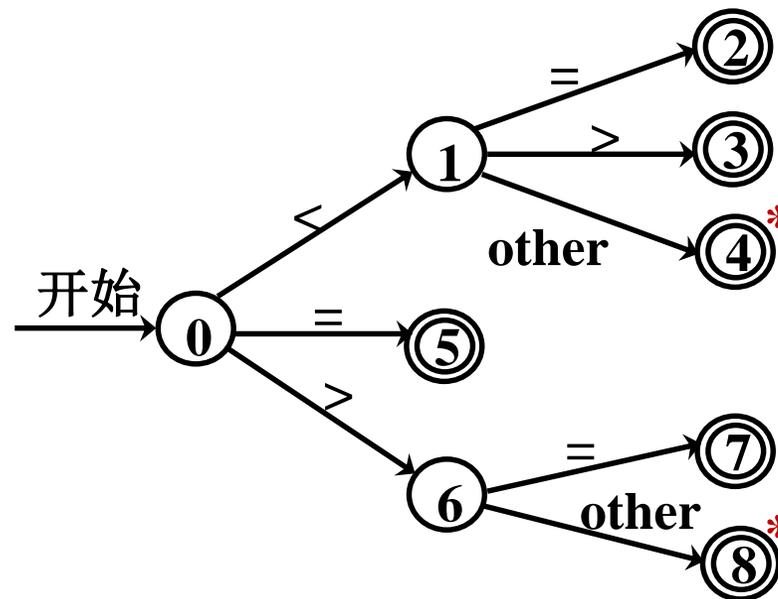
```

TOKEN getRelop() {
  TOKEN retToken = new(RELOP);
  while (1) {
    switch (state) {
      case 0: c = nextChar();
        if (c == '<') state = 1;
        else if (c == '=') state = 5;
        else if (c == '>') state = 6;
        else fail();
        break;
      case 1: ...
      ...
      case 8: retract();
        retToken.attribute = GT;
        return(retToken);
    }
  }
}

```

出错处理，要  
能从错误恢复

回退





$R = \text{Whitespace} \mid \text{Integer} \mid \text{Identifier} \mid '+'$

分析 “foo+3”

→ “f” 匹配  $R$ , 更精确地说是 **Identifier**

→ 但是 “fo” 也匹配  $R$ , “foo” 也匹配, 但 “foo+” 不匹配

如何处理输入? 如果

$x_1 \dots x_i \in L(R)$  并且  $x_1 \dots x_k \in L(R)$ , 假设  $i < k$

则可按“**Maximal munch**”规则来选择:

→ 选择匹配  $R$  的最长前缀, 即选择  $x_1 \dots x_k$

最长匹配规则在实现时: **lookahead**, 不符合则回退



# 词法分析：分类的不确定性

**$R = \text{Whitespace} \mid \text{'new'} \mid \text{Integer} \mid \text{Identifier}$**

分析 “**new foo**”

- “**new**” 匹配  **$R$** ，更精确地说是 ‘**new**’
- 但是也匹配 **Identifier**，此时该选哪个？

一般地，如果  $x_1 \dots x_i \in L(R_j)$  和  $x_1 \dots x_i \in L(R_k)$

规则：选择先列出的模式 ( **$R_j$**  如果  $j < k$ )

- 必须将 ‘**new**’ 列在 **Identifier** 的前面



## 词法分析器对源程序采取非常局部的观点

- 例：难以发现下面的错误

`fi (a == f (x) ) ...`

- 在实数是“数字串.数字串”格式下，可以发现下面的错误

`123.x`

注：数字串长度不小于1

- 紧急方式的错误恢复

删掉当前若干个字符，直至能读出正确的记号

- 错误修补

进行增、删、替换和交换字符的尝试



# 例题 1

写出语言“所有相邻数字都不相同的非空数字串”的正规定义。

**123031357106798035790123**

解答：

$answer \rightarrow (0 \mid no\_0 \ 0) (no\_0 \ 0)^* (no\_0 \mid \varepsilon) \mid no\_0$

$no\_0 \rightarrow (1 \mid no\_0-1 \ 1) (no\_0-1 \ 1)^* (no\_0-1 \mid \varepsilon) \mid no\_0-1$

...

$no\_0-8 \rightarrow 9$

将这些正规定义逆序排列就是答案



## 例题2

下面C语言编译器编译下面的函数时，报告

**parse error before 'else'**

```
long gcd(p,q)
long p,q;
{
    if (p%q == 0)
        /* then part */
        return q      此处遗漏了分号
    else
        /* else part */
        return gcd(q, p%q);
}
```



## 例题2

现在少了第一个注释的结束符号后，反而不报错了

```
long gcd(p,q)
long p,q;
{
    if (p%q == 0)
        /* then part
        return q
    else
        /* else part */
        return gcd(q, p%q);
}
```