北京大学计算机学院

2024年秋季学期　　《编译原理》

第3章 词法分析（1）

Lexical Analysis

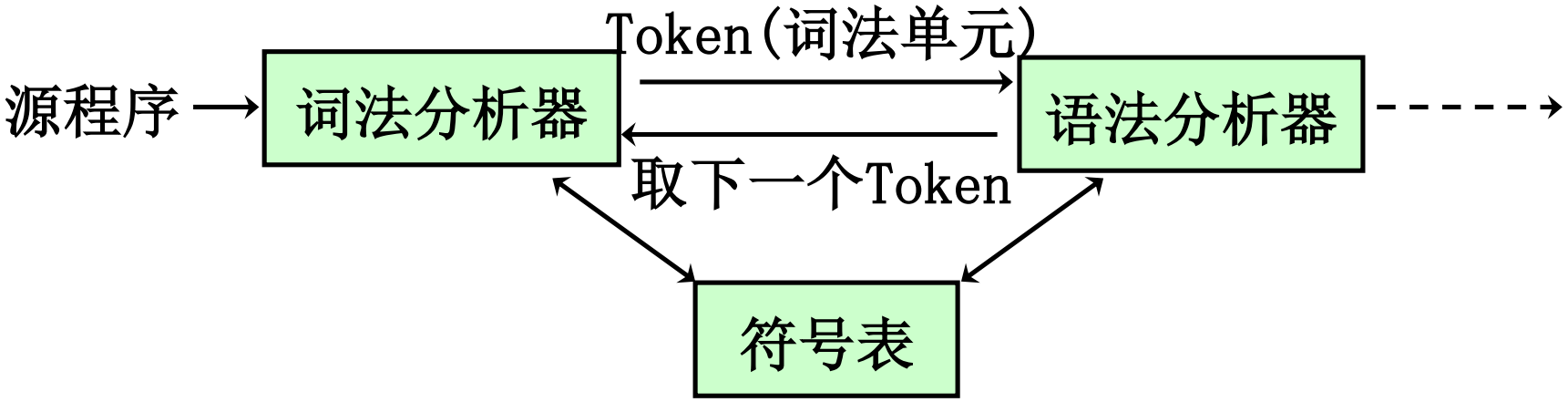
【对应教材2.6, 3.1, 3.2】

内容提要

- 词法分析器的作用
- 词法单元的规约
  - 串和语言；正则表达式、正则定义
- 词法单元的识别
- 词法分析器生成工具—LEX
- 有限自动机 (Finite Automata)
- 正则表达式到有限自动机
- 词法分析器生成工具的设计

词法分析器的作用

- 读入源程序字符流、输出token序列
- 过滤空白、换行、制表符、注释等
- 将token信息添加到符号表中
- 逻辑上独立于语法分析，但是通常和语法分析器处于同一Pass中

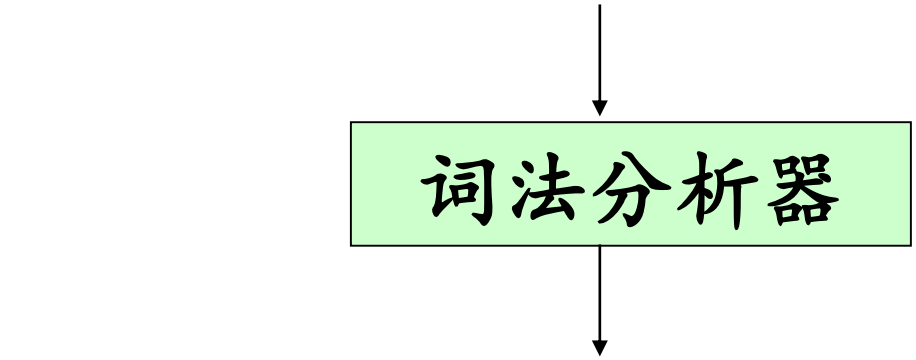


词法单元、模式、词素

- 词法单元（token）
  - <词法单元名、属性值（可选）>
  - 单元名是表示词法单位种类的抽象符号；语法分析器通过单元名即可确定词法单元序列的结构
  - 属性值通常用于语义分析之后的阶段
- 模式（pattern）
  - 描述了一类词法单元的词素可能具有的形式
- 词素（lexeme）
  - 源程序中的字符序列
  - 它和某个词法单元的模式匹配，被词法分析器识别为该词法单元的实例。

词法分析程序的功能举例

```
while  i<>j  do
    if  i>j  then  i:=i-j  else  j:=j-i
```




```
'while', 'i', '<>', 'j', 'do',
'if', 'i', '>', 'j', 'then', 'i', ':=',
'i', '-', 'j', 'else', 'j', ':=', 'j', '-', 'i'
```

词法分析器的功能

- 主要功能：识别词法单元(token)
  - 输出词法单元，包括其名称及相关属性
  - 对数字常数完成数字字符串到二进制数值的转换
  - 去除注释与空白符号
  - 把编译器生成的错误信息关联到源文件
    - 例如：记录每一个符号的位置（行、列）
  - 可能要进行一些预处理工作
    - 比如识别宏（macro），宏的扩展

Token的类别和属性

1. 关键字(保留字) (Keyword)
   - begin, end, if, then, for, while …
2. 标识符 (Identifier)
   - 用来表示各种名字
3. 字面常数 (Literal)
   - 256，3 .14159，1E+3，true，"abc"
4. 运算符 (Operator)
   - 如 十、一、*、/ 等
5. 分界符 (Delimiter)
   - 如逗号，分号，冒号等

词法分析器的输出

- Token的基本输出格式：

  <类别编码，词法单元自身的属性值>

  其中，类别提供给语法分析程序使用；词法单元自身的属性值提供给语义分析程序使用

- 具体的分类设计以方便语法分析程序使用为原则
  - 关键字：可统归为一类，也可以一个关键字分成一类
  - 常数：可统归为一类，也可按类型（整型、实型、布尔型等），每个类型的常数划分成一类
  - 属性值：词法单元属性值的内容，是由词法分析和语义分析的任务划分决定的

词法分析示例-1

```
while  i<>j  do
    if  i>j  then  i:=i-j  else  j:=j-i
```

上述源程序经词法分析器的输出：

〈while，——〉
〈id，指向i的符号表入口的指针〉
〈relational-op，NE〉
〈id，指向j的符号表入口的指针〉
〈do，——〉
〈if，——〉
〈id，指向i的符号表入口的指针〉
……
〈id，指向j的符号表入口的指针〉

词法分析示例-2

- **FORTRAN语言**
  - 空格是没有含义的，例如

```
VAR1 等同于 VA R1
```

  - 对比如下两段代码：

```
DO 5 I = 1,25

DO 5 I = 1.25
```

DO循环语句

赋值语句
DO5I = 1.25

在从左向右分析的过程中，有时候需要**向前看（Lookahead）**。

词法分析示例-3

- PL/1语言
  - 关键字不是保留字

```
IF ELSE THEN THEN = ELSE; ELSE ELSE = THEN
```

  - 声明 vs. 数组

```
DECLARE (ARG1,. . ., ARGN)
```

在词法分析过程中，有时候需要无限长的向前看。

词法分析示例-4

- C++ 语言
  - C++ Template

```
Foo<Bar>
```

  - C++ 流输出语句

```
cin >> var;
```

  - 下面的语句该怎么分析？

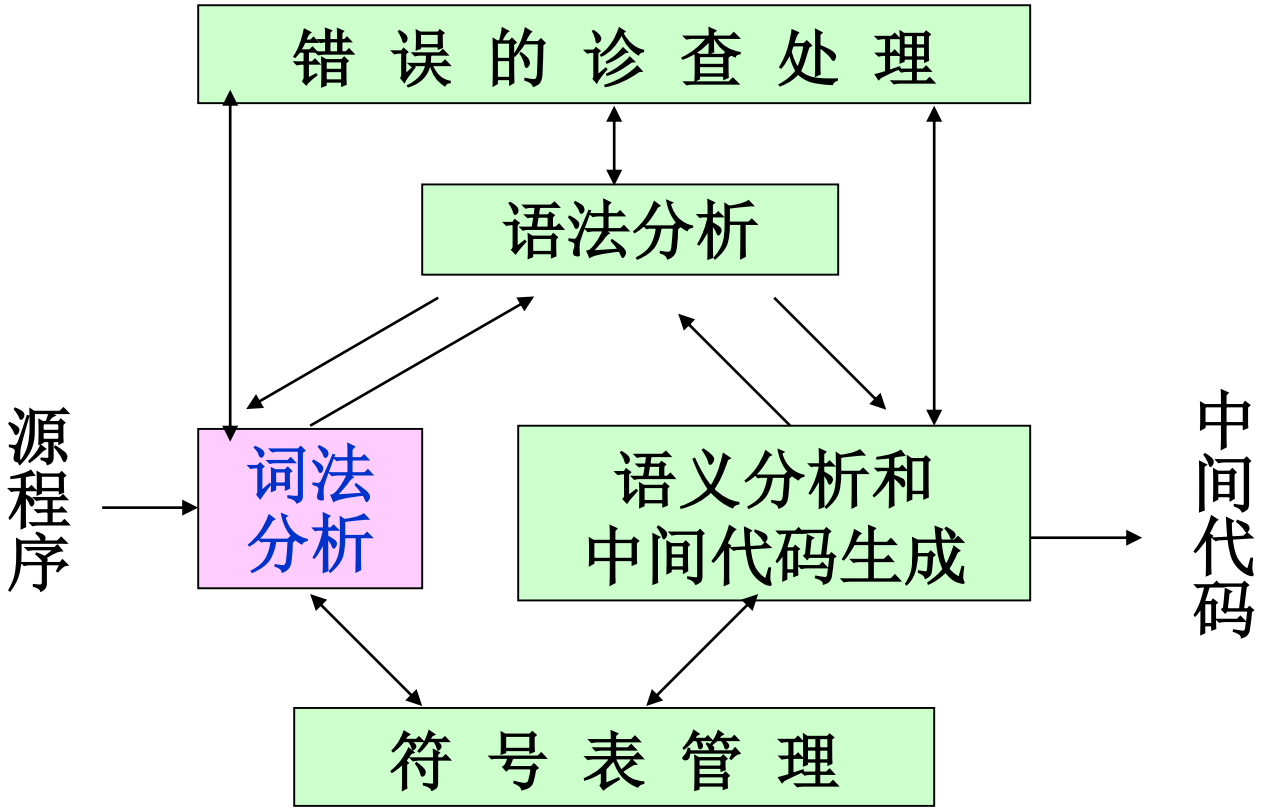
```
Foo<Bar<Barr>>
```

把词法分析设计成一个独立程序

（1）词法分析可以实现为单独的一遍扫描(pass)

（2）词法分析也可以作为语法分析和语义分析的子程序

（每调用一次getToken()函数可以获得一个token）



内容提要

- 词法分析器的作用
- 词法单元的规约
  - 串和语言；正则表达式、正则定义
- 词法单元的识别
- 词法分析器生成工具—LEX
- 有限自动机 (Finite Automata)
- 正则表达式到有限自动机
- 词法分析器生成工具的设计

词法单元的规约

- 规约（Specification）
- 可以用正则表达式来高效、简洁地描述处理词法单元时用到的模式类型。
- 主要内容
  - 字母表
  - 符号串和语言
  - 正则表达式
  - 正则定义
  - 正则表达式的扩展

字母表（Alphabet）

- 字母表是符号的非空有穷集合。任何程序语言都有自己的字母表，例如：
  - 机器语言的字母表：由符号"0"和"1"组成：

    $\Sigma = \{0,1\}$
  - ASCII 字符集
  - Pascal 语言的字母表为：

    $\Sigma = \{A\sim Z, a\sim z, 0\sim 9, +, -, *, /, <, =, >,$
    $:, ', ', ;, ,, ., ^, (,), \{, \}, [,]\}$
  - 也可以类比为自然语言的字母表

符号串（String）

已知字母表Σ，

(1) ε是Σ上的一个符号串（空串）；

(2) 若α是Σ上的符号串，而a是Σ的元素，则αa是Σ上的符号串。

(3) β是Σ上的符号串，当且仅当它由(1)和/或(2)导出（递归定义）。

由字母表中的符号所组成的任意有穷序列被称为该字母表上的符号串（String），也称作“字（Word）”。

通常约定（Convention）

- 用英文字母表比较靠前小写英文字母表示符号
  - 如：a, b, c, d, … ,
- 用小写希腊字母或字母表中靠后的英文字母表示符号串
  - 如：α, β, γ, δ, θ, π, ρ等
    - ε通常用来表示空串
  - 如：……, x, y, z
- 用大写英文表示符号串集合
  - 如：A, B, C, D, … , X, Y, Z

相关术语

设x是一个符号串

- 前缀(prefix)：移走x尾部的零个或多个连续的符号
- 后缀(suffix)：移走x头部的零个或多个连续的符号
- 子串(substring)：从x中删去一个前缀和一个后缀
- 子序列(subsequence)：从x中删去零个或多于零个符号(这些符号不要求是连续的)
- 逆转(reverse, 或称转置，用x^R表示)：将x中的符号按相反次序写出而得到的符号串
- 长度(length)：符号串中的符号的数目

  例如，|aab|=3, |ε|=0

例： 符号串 x=banana

- 前缀：ε, b, ba, ban, bana, banan, banana
- 后缀：banana, anana, nana, ana, na, a, ε
- 子串： banana, anana, banan, anan, …, ε
- 真前缀，真后缀，真子串：y≠x & y≠ε
- 子序列：baa (这些符号不要求是连续的)
- 逆转（用x^R表示）：ananab
- 长度：|banana|=6

符号串的运算

1. 连接(concatenation)：设 x 和 y 是符号串，它们的连接 xy是把 y 的符号写在 x 的符号之后得到的符号串。

例如， x = ba, y = nana, xy = banana.

2. 方幂(exponentiation)： $x^0=\varepsilon$ ； $x^1=x$； $x^2=xx$；

……； $x^n=x^{n-1}x$

例如, x=ba,

x^1= ba, x^2=baba, x^3=bababa,…...

语言（符号串集合）

- 语言（language）：某个给定字母表上一个任意的可数的符号串集合

- 语言的例子
  - 空集∅
  - 只包含空串的集合{ε}
  - 所有符合规范的C语言标识符的集合
  - 所有语法正确的C语言程序的集合
  - 所有语法正确的英语句子的集合

语言的运算

设L和M是两个符号串集合，则：

1. 合并(union)：$L \cup M = \{ s \mid s \in L \text{ 或 } s \in M \}$
2. 连接(concatenation)：$LM = \{ st \mid s \in L \text{ 且 } t \in M \}$
3. 方幂(exponentiation)：$L^0 = \{\varepsilon\}$，$L^1 = L$，
   $L^2 = LL, \ldots, L^n = L^{n-1}L$
4. 语言L的Kleene闭包(closure)，记作L^*，
   $L^* = \cup L^i (i \geq 0) = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \ldots$
5. 语言L的正闭包(positive closure)，记作L^+
   $L^+ = L\,L^*$
   $L^+ = \cup L^i (i \geq 1) = L^1 \cup L^2 \cup L^3 \cup L^4 \cup \ldots$

示例

例：L={A~Z, a~z} D={0~9}

1. L∪D={A~Z, a~z, 0~9}
2. LD 是由所有一个字母后跟一个数字组成的符号串所构成的集合。
3. L^4 是由所有的四个字母的符号串构的集合。
4. $L(L∪D)^*$ 是由所有的字母打头的字母和数字组成的符号串所构成的集合。
5. D^+ 是由所有的长度大于等于1的数字串所构成的集合。

问题

- LM = ML?
- {ε}M = M {ε} ?
- { }M = M{ } ?

正则表达式与正则语言

- 以递归的方式来定义某个字母表∑上的正则表达式及其对应的正则集合（正则语言）
  1. ε是一个正则表达式，表示的语言 L(ε) = {ε}
  2. 如 $a \in \sum$，a 是一个正则表达式，L(a) = {*a*}
  3. 归纳步骤：设 r 和 s 是∑上的正则表达式
     a) (r)|(s)是一个正则表达式，表示语言 L(r) ∪ L(s)
     b) (r)(s) 是一个正则表达式，表示语言 L(r) L(s)
     c) (r)* 是一个正则表达式，表示语言 (L(r))*
     d) (r) 是一个正则表达式，表示语言 L(r)
- 注意：去掉一个正则表达式中的冗余括号之后，它表示的正则语言不变 （注意运算的优先级！）

正则表达式示例

例: $\Sigma=\{a,b\}$

(a)	a | b	$\{a, b\}$
(b)	(a | b) (a | b)	$\{aa, ab, ba, bb\}$
(c)	a*	$\{\varepsilon, a, aa, aaa, aaaa,\ldots\}$
(d)	(a | b)* 或 (a*b*)*	$\{\varepsilon, a, b, aa, ab, ba, bb, aaa,\ldots\}$
(e)	a*b	$\{b, ab, aab, aaab,\ldots\ldots\}$

C 语言标识符:

(A|B|...|Z|a|b|...|z|_) ((A|B|...|Z|a|b|...|z|_) | (0|1|...|9))*

有符号整数:

(+|-|ε) (0|1|...|9) (0|1|...|9)*

Quiz: 选择题

- 下面的正则表达式对应的语言中包含多少个不同的字符串？

$$(0|1|\varepsilon)(0|1|\varepsilon)(0|1|\varepsilon)(0|1|\varepsilon)$$

a) 12
b) 16
c) 31
d) 32
e) 64
f) 81

正则表达式的性质

- 设e_1, e_2和e_3均是某字母表上的正则表达式，则有：
  - 单位正则表达式 ε： $\varepsilon e = e\varepsilon = e$
  - 交换律: $e_1 | e_2 = e_2 | e_1$
  - 结合律: $e_1 | (e_2 | e_3) = (e_1 | e_2) | e_3$

    $e_1(e_2e_3) = (e_1e_2)e_3$
  - 分配律: $e_1(e_2 | e_3) = e_1e_2 | e_1e_3$

    $(e_1 | e_2)e_3 = e_1e_3 | e_2e_3$

此外: $r^* = (r|\varepsilon)^*$ $r^{**} = r^*$

$(r|s)^* = (r^*s^*)^*$

正则定义（Regular Definition）

- 正则定义是如下形式的定义序列：

$$D_1 \rightarrow R_1$$
$$D_2 \rightarrow R_2$$
$$\vdots$$
$$\vdots$$
$$D_n \rightarrow R_n$$

限定：在R_i中只能出现字母表Σ中的字符，以及前面已定义的正则表达式名字，我们用这种辅助定义式(相当于规则)来定义程序语言的单词符号。

- 其中：
  - $R_1, R_2, \ldots\ldots, R_n$ 为正则表达式。
  - $D_1, D_2, \ldots\ldots, D_n$ 为正则表达式名字。

正则定义示例

例：C语言标识符的定义：

$$letter\_ \rightarrow A \mid B \mid \cdots \mid Z \mid a \mid b \mid c \cdots \mid z \mid \_$$

$$digit \rightarrow 0 \mid 1 \mid \ldots \mid 9$$

$$id \rightarrow letter\_ \; (\; letter\_ \mid digit \;) \; *$$

正则表达式的扩展形式

- 为了表达的方便，通常可以对正则表达式做如下的扩展：
  - 1次或多次出现：(r)+ 用来表示 (L(r))+
    - r* = r+ | ε; r+ = rr* = r*r
  - 0次或1次出现：r? 用来表示 r |ε
    - 也就是 L(r) U {ε}
  - 字符类：[abc] 表示 a|b|c；[a-z] 表示 a|b|c|…|z

例：C语言标识符的定义可表示为：

[A-Za-z_] ([A-Za-z_0-9])*

Lex中的正则表达式（图3-5）

表达式	匹配符号	举例
c	单个非运算符字符 c	a
\c	转义符：字符c的字面值(literal value)	*
"s"	串s的字面值	"**"
.	除换行符之外的任何字符	a.*b
^	一行的开始	^abc
$	一行的结尾	abc$
[s]	字符类（字符串s中的任一个字符）	[abc]
[^s]	补集字符类（不在串s中的任一个字符）	[^abc]
r{m, n}	最少m个，最多n个r的重复出现	a{1,5}
r_1 / r_2	后面跟着r_2的r_1	abc/123

与之前讲的一样：r*, r+, r? (重复） r_1 r_2 （连接） r_1 | r_2 （或）

Lex中的正则表达式（2）

- 特殊字符：\ " . ^ $ [ ] * + ? { } | /
  - 双引号串 "s" 可以取消s的特殊含义
  - 转义字符（\）：取消单个字符的特殊含义
  - 例： "**" 和 ** 都可以匹配两个星号 (**)
  - 如何匹配字符串 " \ ?
- 补集字符类（^）：代表该字符类中列出的字符之外的所有字符。
  - 例：[^A-Za-z] 匹配所有非字母的字符
  - 证明：对于每个含有补集字符类的正则表达式,都存在一个等价的不含补集字符类的正则表达式

Lex中的正则表达式（3）

- 运算符 ^ 匹配行开始（一行的最左端）；
  运算符 $ 匹配行结尾（一行的最右端）

  - 例：^[^aeiou]*$ （它匹配的是什么？）

  - 运算符 ^ 还可用于表示补集字符类，如何识别 符号 ^ 的确切含义？会不会有二义性？

  - 是否总是能把包含^和$的正则表达式替换为等价的不含^和$的正则表达式？

例 题 1

- 写出语言“所有相邻数字都不相同的非空数字串”的正则定义。

作业

注：在括号中标注“本”的为教材《本科教学版》对应的习题编号。

（9月20日交作业）

- Ex. 3.3.2 (本Ex. 3.2.2)
- Ex. 3.3.3 (本Ex. 3.2.3)
- Ex. 3.3.5 (本Ex. 3.2.5) (1,2,3,6,9小题)