



# 期末复习大纲

(内部资料)



复习重点:

教材、讲义例题 (含课堂练习)、课后作业

题型: 综合题 (理解 + 应用)

# 第3章 词法分析

## 第 3 章 词法分析 /53

### 3.1 词法分析程序与语法分析程序的关系 /53

### 3.2 词法分析程序的输入与输出 /54

#### 3.2.1 输入缓冲区 /54

#### 3.2.2 词法分析程序的输出 /56

### 3.3 记号的描述和识别 /57

#### 3.3.1 词法与正规文法 /58

#### 3.3.2 记号的文法 /58

#### 3.3.3 状态转换图与记号的识别 /61

### 3.4 词法分析程序的设计与实现 /62

#### 3.4.1 文法及状态转换图 /63

#### 3.4.2 词法分析程序的构造 /65

#### 3.4.3 词法分析程序的实现 /65

### 3.5 LEX 简介 /71

#### 3.5.1 LEX 源程序的结构 /71

#### 3.5.2 LEX 源程序举例 /74

### 习题 3 /76

### 程序设计 1 /77

# 第3章 词法分析

3.2 试用文字描述由下列正规表达式所表示的语言。

✓(1)  $0(0|1)^*0$

(2)  $((\epsilon|0)1^*)^*$

(3)  $(0|1)^*0(0|1)(0|1)$

✓(4)  $0^*10^*10^*10^*$

(5)  $(00|11)^*((01|10)(00|11)^*(01|10)(00|11)^*)^*$

3.3 写出下列各语言的正规表达式。

(1) 处于  $/^*$  和  $^*/$  之间的串构成的注释, 注释中没有  $^*/$ , 除非它们出现在双引号中。

(2) 所有不含子串 011 的由 0 和 1 构成的符号串的全体。

(3) 所有不含子序列 011 的由 0 和 1 构成的符号串的全体。

(4) 以 a 开头和结尾的所有小写字母串。

(5) 所有表示偶数的数字串。

✓ 3.4 构造一文法, 使其语言是无符号偶整数的集合。

(1) 假设允许无符号偶整数以 0 打头。

(2) 假设不允许无符号偶整数以 0 打头。

3.5 请写出 C 语言的字母表。

3.6 请说明 C 语言中定义了哪些记号? 分别给出这些记号的正规表达式和右线性文法。

3.7 画出识别 C 语言关键字 case、char、const 和 continue 的 DFA。

✓ 3.8 C 语言规定其程序中的注释可以有单行和多行两种不同的格式, 单行注释出现在行尾, 其格式形如  $//\dots$ , 多行注释格式形如  $/*\dots*/$ , 请给出一个可以识别这两种风格的注释的 DFA  $D$ 。

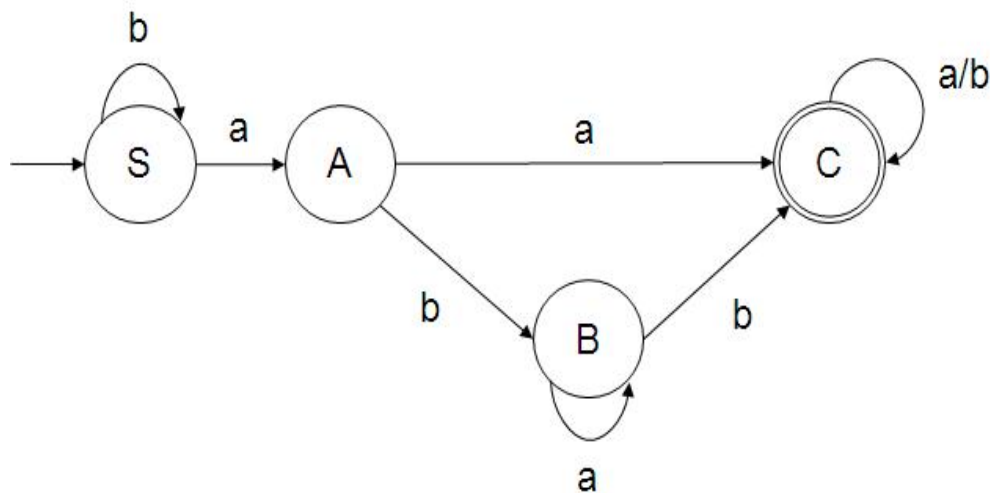
# 课堂练习1

- 自动机 M 的状态转换矩阵如下所示，其中初态是 S，终态是 C。

- (1) 画出相应的状态转换图；
- (2) 写出与之等价的右线性文法。

	a	b
S	A	S
A	C	B
B	B	C
C	C	C

- 解答：



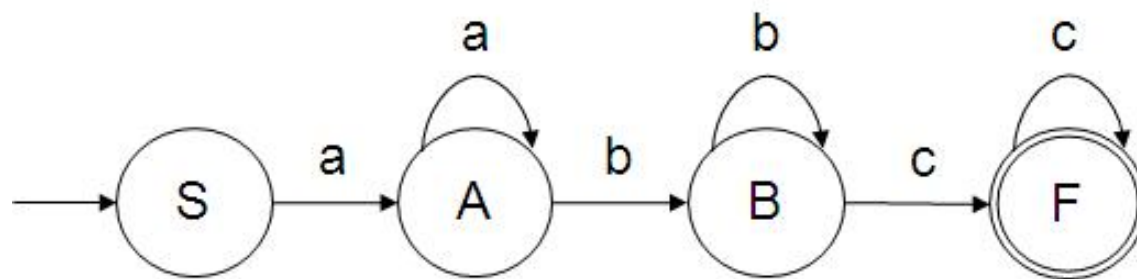
$S \rightarrow aA \mid bS$   
 $A \rightarrow aC \mid bB$   
 $B \rightarrow aB \mid bC$   
 $C \rightarrow aC \mid bC \mid \varepsilon$

## 课堂练习2

■ 自动机 M 的状态转换图如下所示。

(1) 该自动机识别的语言是什么？

(2) 给出与之等价的右线性文法。



解答：

(1) 根据自动机知其产生的语言是： $L=\{a^m b^n c^i \mid m, n, i \geq 1\}$

(2) 与之等价的右线性文法是：

$S \rightarrow aA$

$A \rightarrow aA \mid bB$

$B \rightarrow bB \mid cF$

$F \rightarrow cF \mid \varepsilon$

或者： $S \rightarrow aA$

$A \rightarrow aA \mid bB$

$B \rightarrow bB \mid cF \mid c$

$F \rightarrow cF \mid c$

# 课堂练习3

- 已知正则表达式： $(a^*|b)^*(c|d)$ ，判断下面哪几个正则表达式与其等价，请简述理由。

(1)  $a^*(c|d)|b(c|d)$

(2)  $a^*(c|d)^*|b(c|d)^*$

(3)  $a^*(c|d)|b^*(c|d)$

(4)  $(a|b)^*c|(a|b)^*d$

(5)  $(a^*|b)^*c|(a^*|b)^*d$

- 解答：  
(1)、(2)、(3)与所给正则表达式不等价；  
(4)和(5)与所给正则表达式等价。

# 第4章 语法分析

- 4.1.2 常用的语法分析方法 /78
- 4.1.3 语法错误的处理 /79
- 4.2 自顶向下分析方法 /80
  - 4.2.1 递归下降分析 /81
  - 4.2.2 递归调用预测分析 /82
  - 4.2.3 非递归预测分析 /88
- 4.3 自底向上分析方法 /95
  - 4.3.1 规范归约 /97
  - 4.3.2 “移进-归约”方法的实现 /98
- 4.4 LR 分析方法 /100
  - 4.4.1 LR 分析程序的模型及工作过程 /100
  - 4.4.2 SLR(1)分析表的构造 /104
  - 4.4.3 LR(1)分析表的构造 /112
  - 4.4.4 LALR(1)分析表的构造 /119
  - 4.4.5 LR 分析方法对二义文法的应用 /124
  - 4.4.6 LR 分析的错误处理与恢复 /129
- 4.5 软件工具 YACC /131
  - 4.5.1 YACC 源程序 /132
  - 4.5.2 YACC 对二义文法的处理 /134
  - 4.5.3 用 LEX 建立 YACC 的词法分析程序 /136
- 习题 4 /137
- 程序设计 2 /141

# 第4章 语法分析

✓ 4.3 有文法  $G: A \rightarrow (A)A | \epsilon$

(1) 构造非终结符号  $A$  的 FIRST 集合和 FOLLOW 集合。

(2) 说明该文法是 LL(1) 文法。

✓ 4.5 考虑如下文法  $G$ :

$E \rightarrow A | B$

$A \rightarrow \text{num} | \text{id}$

$B \rightarrow (L)$

✓ 4.9 考虑如下文法  $G$ :

$S \rightarrow AS | b$

$A \rightarrow SA | a$

(1) 构造该文法的 LR(0) 项目集规范族及识别其所有活前缀的 DFA。

(2) 该文是 SLR(1) 文法吗? 为什么?

(3) 构造该文法的 LR(1) 项目集规范族, 该文法是 LR(1) 文法吗?

✓ 4.14 证明下面的文法是 LL(1) 文法, 但不是 SLR(1) 文法。

$S \rightarrow AaAb | BbBa$

$A \rightarrow \epsilon$

$B \rightarrow \epsilon$

✓ 4.16 下面的文法属于哪一类 LR 文法? 试构造其分析表。

$S \rightarrow (SR | a$

$R \rightarrow ,SR | )$



# FOLLOW集合及其构造

## ■ FOLLOW集合

定义：假定S是文法G的开始符号，对于G的任何非终结符号A，集合FOLLOW(A)是在所有句型中，紧跟A之后出现的终结符号或\$组成的集合。

描述为： $\text{FOLLOW}(A) = \{ a \mid S \xRightarrow{*} \dots Aa \dots, a \in V_T \}$

特别地，若 $S \xRightarrow{*} \dots A$ ，则规定 $\$ \in \text{FOLLOW}(A)$

# 构造每个非终结符号A的集合FOLLOW(A)

- 对文法开始符号S，置\$于FOLLOW(S)中，\$为输入字符串的右尾标志。
- 若 $A \rightarrow \alpha B \beta$ 是产生式，则把FIRST( $\beta$ )中的所有非 $\epsilon$ 元素加入到FOLLOW(B)中。
- 若 $A \rightarrow \alpha B$ 是产生式，或 $A \rightarrow \alpha B \beta$ 是产生式并且 $\beta \xRightarrow{*} \epsilon$ ，则把FOLLOW(A)中的所有元素加入到FOLLOW(B)中。

$$S \xRightarrow{*} \dots A a \dots \Rightarrow \dots \alpha B a$$

即：“紧跟在A之后”的终结符都可以“紧跟在B之后”。

- 重复此过程，直到所有集合不再变化为止。

# 第5章 语法制导翻译技术

## 第 5 章 语法制导翻译技术 /142

### 5.1 语法制导定义及翻译方案 /143

#### 5.1.1 语法制导定义 /143

#### 5.1.2 依赖图 /146

#### 5.1.3 计算次序 /147

#### 5.1.4 S 属性定义及 L 属性定义 /148

#### 5.1.5 翻译方案 /149

### 5.2 S 属性定义的自底向上翻译 /151

#### 5.2.1 为表达式构造语法树的语法制导定义 /151

# 第5章 语法制导翻译技术

✓ 5.1 根据表 5-1 中的语法制导定义,为表达式  $(4 * 7 + 1) * 2$  建立一棵注释分析树。

5.2 考虑如下文法,写出对该文法产生的表达式求值的语法制导定义。

$$E \rightarrow TE'$$

$$E' \rightarrow +TE' \mid -TE' \mid \varepsilon$$

$$T \rightarrow FT'$$

$$T' \rightarrow *FT' \mid \varepsilon$$

$$F \rightarrow (E) \mid \text{num}$$

5.3 根据表 5-4 中的语法制导定义为表达式  $((a) + (b))$  建立分析树和语法树。

✓ 5.4 考虑如下的语法制导定义。

产生式	语义规则
$S' \rightarrow S$	$S.u = 5$ $Print(S.v)$
$S \rightarrow ABC$	$B.u = S.u$ $A.u = B.v + C.v$ $S.v = A.v$
$A \rightarrow a$	$A.v = 3 * A.u$
$B \rightarrow b$	$B.v = B.u$
$C \rightarrow c$	$C.v = 2$

(1) 画出字符串  $abc$  的分析树,给出其相应的依赖图。

(2) 根据依赖图,写出一个有效的语义规则执行顺序。

(3) 给出翻译完成时的输出结果。

(4) 如果将上述语法制导定义修改为:

产生式	语义规则
$S' \rightarrow S$	$S.u = 5$ $Print(S.v)$
$S \rightarrow ABC$	$B.u = S.u$ $C.u = B.v$ $A.u = B.v + C.v$ $S.v = A.v$
$A \rightarrow a$	$A.v = 3 * A.u$
$B \rightarrow b$	$B.v = B.u$
$C \rightarrow c$	$C.v = C.u - 2$

则翻译完成时输出的结果值是什么?

# 第5章 语法制导翻译技术

5.5 下面的文法产生对整型数和实型数应用“+”算符形成的表达式。两个整型数相加，

176

第5章 语法制导翻译技术

结果仍为整型；否则为实型数。

$$E \rightarrow E + T \mid T$$
$$T \rightarrow \text{num}, \text{num} \mid \text{num}$$

✓(1) 给出一个确定每个子表达式类型的语法制导定义。

(2) 扩充(1)中的语法制导定义,使之既确定类型,又把表达式翻译为前缀形式。使用一元算符 `inttoreal` 把整型数转换为等价的实型数,使得前缀形式中的“+”作用于两个同类型的运算对象。

# 第5章 语法制导翻译技术

√ 5.8 考虑如下产生 Pascal 声明语句的文法。

$D \rightarrow L : T$

$T \rightarrow \text{integer} | \text{real}$

$L \rightarrow L, \text{id} | \text{id}$

(1) 给出确定变量类型的语法制导定义。

(2) 该定义是  $L$  属性定义吗?

5.9 假定声明由下面的文法产生:

$D \rightarrow \text{id} L$

$L \rightarrow, \text{id} L | : T$

$T \rightarrow \text{integer} | \text{real}$

(1) 试设计一个翻译方案,它把每一个标识符的类型信息加入到符号表中。

(2) 根据(1)的翻译方案构造一个预测翻译程序。

√ 5.10 考虑如下的语法制导定义:

产生式	语义规则
$S \rightarrow B$	$B.ps = 10$
$B \rightarrow B_1 B_2$	$B_1.ps = B.ps$ $B_2.ps = B.ps$ $B.ht = \max(B_1.ht, B_2.ht)$
$B \rightarrow B_1 \text{sub} B_2$	$B_1.ps = B.ps$ $B_2.ps = B.ps$ $B.ht = \text{disp}(B_1.ht, B_2.ht)$
$B \rightarrow \text{text}$	$B.ht = \text{text}, h \times B.ps$

(1) 判断该语法制导定义是否为  $L$  属性定义。

(2) 给出该语法制导定义相应的翻译方案。

(3) 改造(2)所得翻译方案,使之可用 LR 方法进行翻译。

(4) 根据(3)所得翻译方案,设计与各产生式相应的代码段。

(5) 根据(4)所设计代码段,举例说明,每当把一个右部归约为  $B$  时,继承属性  $B.ps$  的值在栈中的位置总是恰好在归约串的下面。

# 第5章 语法制导翻译技术

✓ 5.16 有如下文法：

$$S \rightarrow (L) \mid a$$
$$L \rightarrow L, S \mid S$$

(1) 设计一个语法制导定义，它输出配对的括号个数。

178

第5章 语法制导翻译技术

(2) 构造一个翻译方案，它输出每个  $a$  的嵌套深度。如对句子  $(a, (a, a))$  的输出结果是 1, 2, 2。

✓ 5.17 令综合属性  $val$  给出在下面的文法中  $S$  产生的二进制数的值，如对于输入 101.101：  $S.val = 5.625$

$$S \rightarrow L.L \mid L$$
$$L \rightarrow LB \mid B$$
$$B \rightarrow 0 \mid 1$$

请写出确定  $S.val$  值的语法制导定义。

# 第6章 语义分析

第 6 章	语义分析	/180
6.1	语义分析概述	/180
6.1.1	语义分析的任务	/180
6.1.2	语义分析程序的位置	/181
6.1.3	错误处理	/181
6.2	符号表	/182
6.2.1	符号表的建立和访问时机	/182
6.2.2	符号表内容	/184
6.2.3	符号表操作	/187
6.2.4	符号表组织	/189
6.3	类型检查	/193
6.3.1	类型表达式	/194
6.3.2	类型等价	/197
6.4	一个简单的类型检查程序	/204
6.4.1	语言说明	/204
6.4.2	符号表的建立	/205
6.4.3	表达式的类型检查	/210
6.4.4	语句的类型检查	/213
6.4.5	类型转换	/214



# 第7章 运行环境

## 第 7 章 运行环境 /225

### 7.1 程序运行时的存储组织 /225

#### 7.1.1 程序运行空间的划分 /226

#### 7.1.2 活动记录与控制栈 /227

#### 7.1.3 名字的作用域及名字绑定 /230

### 7.2 存储分配策略 /231

#### 7.2.1 静态存储分配 /231

#### 7.2.2 栈式存储分配 /233

#### 7.2.3 堆式存储分配 /237

### 7.3 非局部名字的访问 /239

#### 7.3.1 程序块 /239

#### 7.3.2 静态作用域规则下非局部名字的访问 /241

#### 7.3.3 动态作用域规则下非局部名字的访问 /248

### 7.4 参数传递机制 /250

#### 7.4.1 传值调用 /250

#### 7.4.2 引用调用 /252

#### 7.4.3 复制恢复 /253

#### 7.4.4 传名调用 /255

## 习题 7 /255

# 第7章 运行环境

√ 7.2 有如下的 C 语言程序,采用下列参数传递方式时的输出分别是什么?

- (1) 传值调用
- (2) 引用调用
- (3) 复制恢复(假定按从左到右的顺序把结果复制回实参)
- (4) 传名调用

```
#include<stdio.h>
int k;
int a[3];
void swap(int x,int y)
{  x=x+y;
   y=x-y;
   x=x-y;
}
void main()
{  k=1;
   a[0]=2;
   a[1]=1;
   a[2]=0;
   swap(k,a[k]);
   printf("k=%d,a[0]=%d,a[1]=%d,a[2]=%d\n",k,a[0],a[1],a[2]);
   swap(a[k],a[k]);
   printf("k=%d,a[0]=%d,a[1]=%d,a[2]=%d\n",k,a[0],a[1],a[2]);
}
```

# 第7章 运行环境

✓ 7.3 假如编译程序采用不同的参数传递方式处理下面的程序,所生成的目标程序在运行时的输出分别是什么?

- (1) 传值调用
- (2) 引用调用
- (3) 复制恢复(假定按从左到右的顺序把结果复制回实参)
- (4) 传名调用

256

第 7 章 运行环境

```
program main(input,output);  
  procedure p(x,y,z);  
    begin  
      y:=y+1;  
      z:=z+x  
    end;  
  begin  
    a:=2;  
    b:=3;  
    p(a+b,a,a);  
    writeln ('a=',a)  
  end.
```

# 第7章 运行环境

√7.5 考虑下面的 Pascal 程序：

```
(1) program main(input, output);  
(2)   procedure b(function h(n: integer): integer);  
(3)       var m: integer;  
(4)       begin m:=3;writeln(h(2)) end;   { end of b }  
(5)   procedure c;  
(6)       var m: integer;  
(7)       function f(n: integer): integer;  
(8)           begin f:=m+n end;           { end of f }  
(9)       procedure r;  
(10)           var m: integer;  
(11)           begin m:=7; b(f) end;   { end of r }  
(12)       begin m:=0; r end;           { end of c }
```

257

《编译原理与技术（第2版）》 第7章

```
(13) begin c end.{ end of main }
```

- (1) 该程序的输出结果是什么？
- (2) 试画出该程序的活动树。(可选)
- (3) 试画出当控制处于函数 f 中时的控制栈状态，要求标出其中的控制链和访问链。

# 第8章 中间代码生成

## 第 8 章 中间代码生成 /259

### 8.1 中间代码形式 /259

#### 8.1.1 图形表示 /259

#### 8.1.2 三地址代码 /260

IX

## 目 录 《编译原理与技术（第 2 版）》

### 8.2 赋值语句的翻译 /265

#### 8.2.1 仅涉及简单变量的赋值语句的 翻译 /265

#### 8.2.2 涉及数组元素的赋值语句 /268

#### 8.2.3 记录结构中域的访问 /273

# 第8章 中间代码生成

8.3 请把语句  $\text{if } (x+y) * z = 0 \text{ then } s := (a+b) * c \text{ else } s := a * b * c$  翻译为：

(1) 语法树。

✓(2) 三地址代码。

8.4 有如下的 C 语言程序片断,请把其中的可执行语句翻译为：

(1) 语法树。

✓(2) 三地址代码。

```
main() {  
    int i;  
    int a[10];  
    i=0;  
    while (i<10) {  
        a[i]=0;  
        i++;  
    }  
}
```

# 第9章 目标代码生成

## 第 9 章 目标代码生成 /297

### 9.1 目标代码生成概述 /297

#### 9.1.1 代码生成程序的位置 /297

#### 9.1.2 代码生成程序设计的相关问题 /298

### 9.2 基本块和流图 /300

### 9.3 下次引用信息 /302

### 9.4 一个简单的代码生成程序 /305

#### 9.4.1 目标机器描述 /305

#### 9.4.2 代码生成算法 /307

#### 9.4.3 其他常用语句的代码生成 /312

### 习题 9 /315

# 第9章 目标代码生成

## 习题 9

✓ 9.1 有如下的三地址代码：

```
1 ✓ read(n)
    i:=1
    fen:=1
4 ✓ L1: if i<=n goto L2
    5 ✓ goto L3
6 ✓ L2: t1:=fen * i
    fen:=t1
    i:=i+1
    goto L1
10 ✓ L3: write(fen)
```

- (1) 将该代码段划分为基本块。
- (2) 基于(1)的结果,构造相应的流图。



# 第10章 代码优化

10.2.1 常数合并及常数传播 /318

10.2.2 删除公共表达式 /320

10.2.3 复制传播 /321

10.2.4 削弱计算强度 /321

10.2.5 改变计算次序 /321

10.3 dag 在基本块优化中的应用 /322

10.3.1 基本块的 dag 表示 /322

10.3.2 基本块的 dag 构造算法 /323

10.3.3 dag 的应用 /327

10.3.4 dag 构造算法的进一步讨论 /330

10.4 循环优化 /333

10.4.1 循环展开 /333

10.4.2 代码外提 /334

10.4.3 削弱计算强度 /334

10.4.4 删除归纳变量 /335

# 第10章 代码优化

✓10.3 有如下三地址代码：

```
— I:=1  
  read J,K  
— L: A:=K*I  
    B:=J*I  
    C:=A*B  
    write C  
    I:=I+10  
    if I<100 goto L  
— halt
```

- (1) 画出它的流图。
- (2) 对这段代码进行优化。

✓10.4 有如下程序段：

```
var a,b: array[1..m,1..n] of integer;  
    i,j: integer;  
for i:=1 to m do  
  for j:=1 to n do  
    a[i,j]:=b[i,j]
```

- (1) 请把该程序段中的可执行语句翻译为三地址代码。
- (2) 将(1)的结果划分为基本块，并画出其流图。
- (3) 对内循环代码进行所有可能的优化，要求依次写出所采用的优化技术、关键步骤，给出每种优化后的结果。

预祝同学们考出优异的成绩！