

# 2021-2022 编译原理卷 A

Garone Lombard

2023 年 12 月 13 日

## 摘要

编译原理 2021-2022 试卷答案自行整理

目 录	3
-----	---

## 目录

1 填空	4
2 正则文法与自动机	5
3 LL(1) 和算符优先分析法	7
4 SLR 分析法	8
5 符号表构造与运行时存储分析	11
6 代码优化	14

## 1 填空

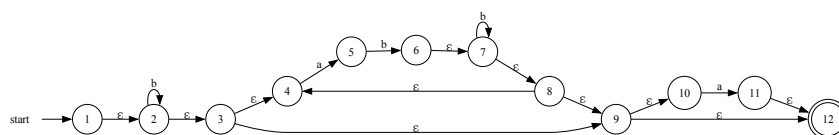
1. 编译过程本质上是 程序转换/翻译 过程，将用 高级语言 书写的源程序加工为与其等价的目标程序。
2. 在编译过程的 5 个基本阶段都要做 符号表管理 和 错误处理 两件事，因此典型的编译程序常划分为 7 个逻辑组成部分
3. 对源程序 (包括源程序中间形式) 从头到尾扫描一遍，并做有关的加工处理，生成新的源程序中间形式或目标程序，通常称之为 一遍，完成编译工作最少需要对源程序做 1 次扫描
4. 生产中间代码的目的是便于做 代码优化 和 编译程序移植
5. 有文法规则  $S \rightarrow if\ E\ S \mid if\ E\ S\ else\ S$ ，用扩充的 BNF 范式表示为  $if\ E\ S\ [else\ S]$
6. 常见的程序设计语言按乔姆斯基的分类是 1 型文法，也称为上下文无关文法。如果采用属性翻译文法处理声明语句 inta; 时，通常可以得到变量类型和名字这样的 继承 属性，并填入到 符号表 中，以便在使用变量 a 时，能够查找到变量的有关信息。没有声明就使用变量，这属于 语义错误，在语法分析只能进行句子的结构分析时并不能发现这个问题
7. 对文法  $G[T]: T \Rightarrow T - T | T / T | (T) | i$ ，规范句型  $T - T / i$  的句柄为 i 和 T-T，由此判断该文法 有 (有/无) 二义性。
8. 规范归约每次归约的是句型的 句柄，算符优先分析法每次归约的是当前句型的 最左素短语
9. 活动记录中 Display 区存放的是 各外层模块活动记录的基地址
10. 文法  $G = (V_n, V_t, P, Z)$ ，其中  $V_t$  代表 文法中的终结符集合

## 2 正则文法与自动机

题干 有如下正则表达式

$$b * (abb *) * (a|\epsilon)$$

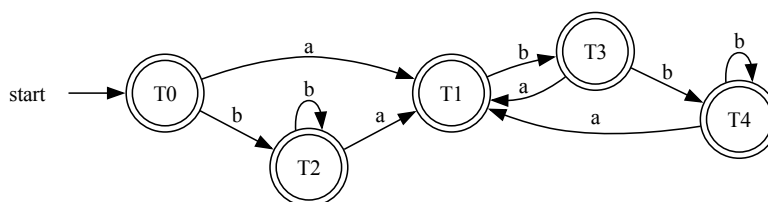
1. 根据正则表达式构造 NFA



2. 将得到的 NFA 确定化

$\epsilon - \text{closure}(\text{start})$	{1,2,3,4,9,10,12}	
输入	a	b
状态		
T0={1,2,3,4,9,10,12}	T1={5,11,12}	T2={2,3,4,9,10,12}
T1={5,11,12}	$\phi$	T3={4,6,7,8,9,10,12}
T2={2,3,4,9,10,12}	T1={5,11,12}	T2={2,3,4,9,10,12}
T3={4,6,7,8,9,10,12}	T1={5,11,12}	T4={4,7,8,9,10,12}
T4={4,7,8,9,10,12}	T1={5,11,12}	T4={4,7,8,9,10,12}

表 1: 转换表



## 3. 将得到的 DFA 最小化

状态 \ 输入	a	b
0	1	2
1	$\phi$	3
2	1	2
3	1	4
4	1	4

表 2: 转换表

显然没有无用状态

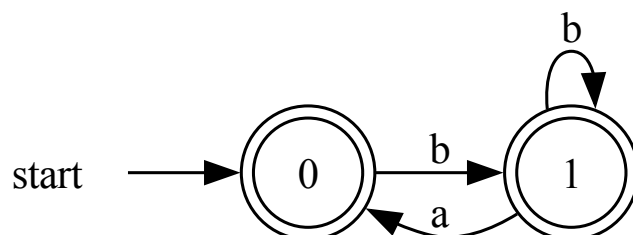
首先将状态分为终态和非终态两个集合  $\{0,1,2,3,4,5\}, \phi$

接下来考察  $\{0,1,2,3,4,5\}$  是否可分

状态 \ 输入	a	b
0	1	2
1	$\phi$	3
2	1	2
3	1	4
4	1	4

表 3: 转换表

所以可分为  $\{1\}, \{0,2,3,4,5\}$  两个集合



### 3 LL(1) 和算符优先分析法

#### 1. 证明所有二义性文法都不是 LL(1) 文法

**证明** 二义性文法的句型有两个不同的最左推导，即存在两个不同的最左句型，因此在构造 LL(1) 分析表时，会出现同一个非终结符对应两个不同的终结符的情况，因此所有二义性文法都不是 LL(1) 文法

#### 2. 已知文法 G[T]:

$$\begin{aligned}
 T &\rightarrow T - F | F \\
 F &\rightarrow F / P | P \\
 P &\rightarrow (T) | i
 \end{aligned}
 \tag{1}$$

#### 2.1 求各文法的 FIRSTVT 集和 LASTVT 集

FIRSTVT	-	/	(	)	i
T	1	1	1		1
F		1	1		1
P			1		1

表 4: FIRSTVT

LASTVT	-	/	(	)	i
T	1	1		1	1
F		1		1	1
P				1	1

表 5: LASTVT

**2.2** 构造文法  $G$  的优先关系矩阵，并判断该文法是否是算符优先文法

分析表	-	/	(	)	i
-	$\succ$	$\prec$	$\prec$	$\succ$	$\prec$
/	$\succ$	$\succ$	$\prec$	$\succ$	$\prec$
(	$\prec$	$\prec$	$\prec$	$\doteq$	$\prec$
)	$\succ$	$\succ$	err	$\succ$	err
i	$\succ$	$\succ$	err	$\succ$	err

表 6: 分析表

没有重复的  $\prec$  或  $\succ$ ，因此该文法是算符优先文法

## 4 SLR 分析法

题干 有如下文法  $G[S]$ :

$$\begin{aligned}
 S &\rightarrow CD|DC \\
 C &\rightarrow aCb|ab \\
 D &\rightarrow Db|b
 \end{aligned}
 \tag{2}$$

**1.** 拓展文法，使得文法的开始符号仅出现在一个产生式的左侧；求原文法所有非终结符的 FOLLOW 集

“使得文法的开始符号仅出现在一个产生式的左侧”：只需构造文法的增广文法即可，如下所示



$$\begin{aligned}
S' &\rightarrow S \\
S &\rightarrow CD|DC \\
C &\rightarrow aCb|ab \\
D &\rightarrow Db|b
\end{aligned}
\tag{3}$$

集合	FIRST	FOLLOW
S	{a,b}	{#}
C	{a}	{b,#}
D	{b}	{a,b,#}

表 7: LASTVT

2. 求拓展后的 SLR 分析表，包括 GOTO 表和 ACTION 表，表头如下

状态	ACTION			GOTO		
	a	b	#	S	C	D
0	s1	s2		3	4	5
1	s1	s9			8	
2	r7	r7	r7			
3			acc			
4		s2				6
5	s1	s10			7	
6		s10	r2			
7			r3			
8		s11				
9		r5	r5			
10	r6	r6	r6			
11		r4	r4			

3. 求能识别规范句型 aabbbb 活前缀的有效项目集  
句柄: ab  
活前缀: a,aa,aab

a:

$$C \rightarrow a \cdot Cb$$

$$C \rightarrow a \cdot b$$

$$C \rightarrow \cdot aCb$$

$$C \rightarrow \cdot ab$$

(4)

aa:

$$C \rightarrow a \cdot Cb$$

$$C \rightarrow a \cdot b$$

$$C \rightarrow \cdot aCb$$

$$C \rightarrow \cdot ab$$

(5)

aab:

$$C \rightarrow ab \cdot$$

(6)

## 5 符号表构造与运行时存储分析

```
1 program main;
2   var x, y : real;
3   i, k: integer;
4   name: array [1...10] of char;
5   procedure P1 (ind:integer);
6     var x : integer;
7     procedure P2 (j : real);
8       procedure P3;
9         var f : array [1...5] of integer;
10        test1: boolean;
11        begin
12          ...
13        end;{注释:P3}
14      begin
15        P3;
16        ...
17      end;{注释:P2}
18    procedure P4;
19      var r1,r2 : real;
20    begin
21      r1:=y ;
22      r2:=r1+y ;
23      P2(r1+r2);
24      ...
25    end; {注释:P4}
26  begin
27    P4;
28    ...
29  end;{注释:P1}
30 begin
31   P1(100);
```

```
32  ...  
33  end {注释:main}
```

1. 按照以下格式，画出递归下降编译到第 21 行时，栈式符号表的内容

序号	名字	种类	类型	层号
1	x	var	real	1
2	y	var	real	1
3	i	var	integer	1
4	k	var	integer	1
5	name	var	array	1
6	P1	proc		1
7	ind	para	integer	2
8	x	var	integer	2
9	P2	proc		2
10	P4	proc		2
11	r1	var	real	3
12	r2	var	real	3

2. 运行到第 12 行时，运行栈的内容如下所示，将空白处填满

test1	
f	
f 的模板	
prev abp:abp4	
ret addr	
abp4(DISPLAY)	
abp2(DISLPAY)	
adp1(DISPLAY)	abp5: P3
j	
prev abp:abp3	
ret addr	
abp2(DISPLAY)	
abp1(DISPLAY)	abp4: P2
r2	
r1	
prev abp:abp2	
ret addr	
abp2(DISPLAY)	
abp1(DISPLAY)	abp3: P4
x	
ind	
prev abp:abp1	
ret addr	
abp1(DISPLAY)	abp2: P1
name	
name 模板	
k	
i	
y	
x	abp1: main

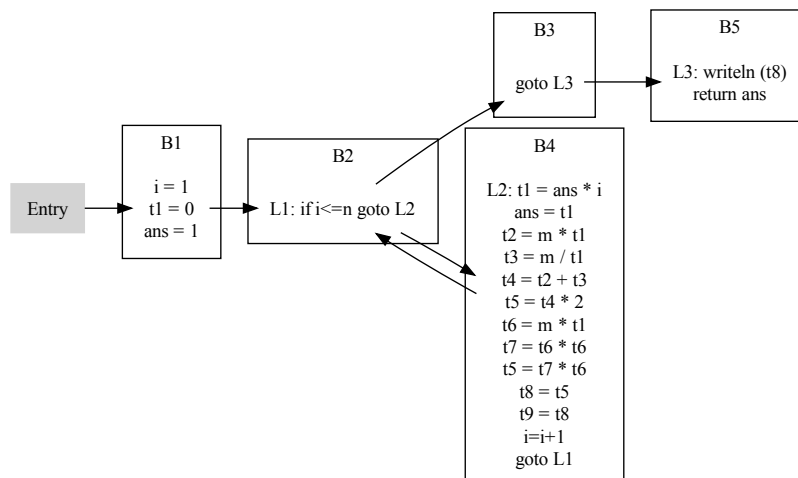
## 6 代码优化

题干 有如下程序，其中  $n, m$  是形参， $i, ans, t1, t2, t3, t4, t5, t6, t7, t8, t9$  都是局部变量

```
1      i = 1
2      t1 = 0
3      ans = 1
4 L1:  if i<=n goto L2
5      goto L3
6 L2:  t1 = ans * i
7      ans = t1
8      t2 = m * t1
9      t3 = m / t1
10     t4 = t2 + t3
11     t5 = t4 * 2
12     t6 = m * t1
13     t7 = t6 * t6
14     t5 = t7 * t6
15     t8 = t5
16     t9 = t8
17     i=i+1
18     goto L1
19 L3:  writeln (t8)
20     return ans
```

1. 将该代码划分基本块，构造相应的控制流图

*i = 1
t1 = 0
ans = 1
*L1: if i <= n goto L2
*goto L3
*L2: t1 = ans * i
ans = t1
t2 = m * t1
t3 = m / t1
t4 = t2 + t3
t5 = t4 * 2
t6 = m * t1
t7 = t6 * t6
t5 = t7 * t6
t8 = t5
t9 = t8
i=i+1
goto L1
*L3: writeln (t8)
return ans



2. 试对 L2 所在的基本块用 DAG 做局部公共子表达式删除优化，并根据启发式算法给出优化后的中间代码序列

OUT[B]	B1	B2	B3	B4	B5
i	1				
ans					
n					
m					
t1					
t2					
t3					
t4					
t5					
t6					
t7					
t8					
t9					



3. 给出每个基本块的 def 和 use 集合，做活跃变量分析，并给出变量的冲突图。注意：变量 A,B 冲突的标准为，变量 B 的定义点处变量 A 活跃，反之亦然