

北京航空航天大学

2019—2020 学年 第一学期期末

《编译技术》

B 卷

班 级_____学 号 _____

姓 名_____成 绩 _____

2020 年 5 月 25 日

班号_____ 学号_____ 姓名_____ 成绩_____

《 编译技术 》期末考试卷

注意事项：

- 1. 试卷共 6 页（不含封面和目录），请仔细检查。
- 2. 在监考老师统计完试卷后，再离开考场；

题目：

- 一、简答题..... (20 分)
- 二、正则文法与自动机..... (20 分)
- 三、算符优先分析法.....(15 分)
- 四、SLR 分析法..... (15 分)
- 五、符号表构造与运行时存储分析..... (15 分)
- 六、代码优化.....(15 分)

题号	得分	教师签字
1		
2		
3		
4		
5		
6		
总分		

编译程序

发现错误 → 限制局部 误差 | 错误扩散
其他部分分析

一. 简答题: (20 分)

1. 简述什么是错误的局部化处理, 主要作用是什么?

错误局部化处理: 指当编译程序发现错误后, 尽可能把错误的影响限制在一个局部的范围, 避免错误扩散和影响 程序其他部分的分析。

根据文法规则, 分析语法成分, 找语法错误。

2. 分别简述语法分析的任务和语义分析的任务。

对分析识别出的 语法成分进行语义分析

语法分析功能: 根据文法规则, 从源程序单词符号串中识别出语法 成分, 并进行语法检查。

基本任务: 识别符号串 S 是否为某语法成分。

找语义错误

语义分析是审查源程序有无语义错误, 为代码生成阶段收集类型信息。

编译

目标程序运行时

编译能力

目标能力

3. 分别简述静态存储分配和动态存储, 二者的使用场景有什么不同。

静态存储分配

变量大小确定且不改变 运行时分配, 编译时生成静态分配指令

在编译阶段由编译程序实现对存储空间的 管理和为源程序中的变量分配存储的方法。

条件

如果在编译时能够确定源程序中变量在运行时的数据空间大小, 且运行时不改变, 那么 就可以采用静态存储分配方法。

动态存储分配

在目标程序运行阶段由目标程序实现对存储空间的组织与管理, 和为源程序中的变量 分配存储的方法。

特点

- 在目标程序运行时进行变量的存储分配。 • 编译时要生成进行动态分配的目标指令。

4. 分别说明什么是局部优化、全局优化和循环优化。

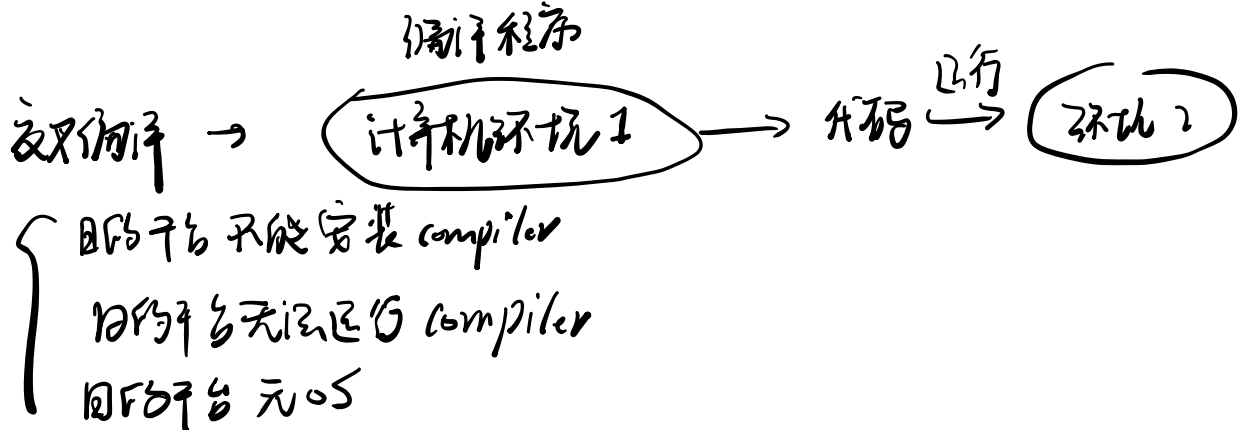
- 局部优化技术
 - 指在基本块内进行的优化
 - 例如，局部公共子表达式删除
- 全局优化技术
 - 函数/过程内进行的优化 – 跨越基本块
 - 例如，全局数据流分析
- 跨函数优化技术 – 整个程序
 - 例如，跨函数别名分析，逃逸分析 等

与循环/循环优化
? 代码外块
循环展开
循环变换

5. 说明什么是交叉编译，什么时候需要使用交叉编译？

在一种计算机环境中运行的编译程序，能编译出在另外一种环境下运行的代码，我们就称这种编译器支持交叉编译。这个编译过程就叫交叉编译。

有时是因为目的平台上不允许或不能够安装我们所需要的编译器，而我们又需要这个编译器的某些特征；有时是因为目的平台上的资源贫乏，无法运行我们所需编译器；有时又是因为目的平台还没有建立，连操作系统都没有，根本谈不上运行什么编译器。



二. 在输入字母表 $\Sigma = \{0, 1\}$ 上定义语言 $L = \{x | x \in \Sigma^*, \text{且 } x \text{ 以 } 0 \text{ 开头, 以 } 1 \text{ 结尾}\}$ 。(20 分)

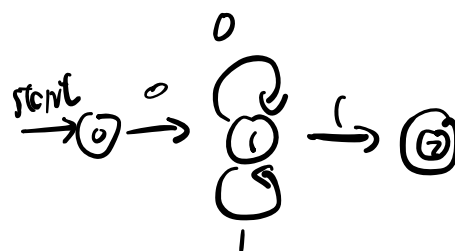
1. 给出识别该语言 L 的正则表达式。

$0(0|1)^*1$

$0(0|1)^*1$

2. 根据正则表达式构造 NFA。

X



3. 设计识别该语言的极小化 DFA。(给出求解过程：①对应的)

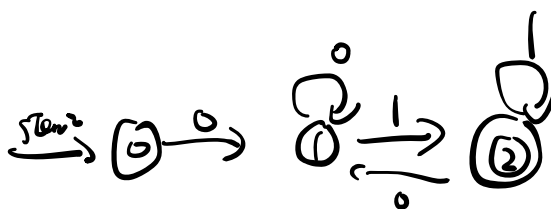
求解

$I - \text{closure}(0) = 0$

Z	Z_0	Z_1
$\{0\}$	$\{1\}$	\emptyset
$\{1\}$	$\{1\}$	$\{1, 2\}$
$\{1, 2\}$	$\{1\}$	$\{1, 2\}$

$0 \rightarrow 1$	$0 \rightarrow \emptyset$
$1 \rightarrow 1$	$1 \rightarrow 1, 2$
$1 \rightarrow 1$	$1 \rightarrow 1, 2$

符	0	1
0	1	-
1	1	2
2	1	2



最小化

0	1
1	-
1	2

已最小化



三. 有如下文法 $G[S]$: (15 分)

$$S \rightarrow V$$

$$V \rightarrow T \mid ViT$$

$$T \rightarrow F \mid T+F$$

$$F \rightarrow bV^* \mid a$$

文法中元 $U ::= \dots VW \dots$

$V, W, \in V_n$

1. 判断该文法是否是算符文法, 并说明理由。

是, 因为文法中无形式如 $U ::= \dots VW \dots$ 的规则

2. 求每个非终结符的 FIRSTVT 和 LASTVT 集合。

$$F(S) = \{i, +, b, a\}$$

$$\text{FIRSTVT}(S) = \text{FIRSTVT}(V) = \{a, b, i, +\}$$

$$L(S) = \{i, +, *, a\}$$

$$\text{LASTVT}(S) = \text{LASTVT}(V) \cup \{i\} = \{a, b, i, +\}$$

$$F(V) = \{i, +, b, a\}$$

$$\text{FIRSTVT}(V) = \text{FIRSTVT}(T) \cup \{i\} = \{a, b, i\}$$

$$L(V) = \{i, +, *, a\}$$

$$\text{LASTVT}(V) = \{a, b\}$$

$$F(T) = \{+, b, a\}$$

$$L(T) = \{+, *, a\}$$

$$\text{LASTVT}(T) = \text{LASTVT}(V) = \{*, a, +, i\}$$

$$F(F) = \{b, a\}$$

$$\text{LASTVT}(V) = \text{LASTVT}(T) \cup \{i\} = \{*, a, +, i\}$$

$$L(F) = \{*, a\}$$

$$\text{LASTVT}(V) = \text{LASTVT}(T) \cup \{i\} = \{*, a, +\}$$

$$\text{FIRSTVT}(F) = \{*, a\}$$

$\# \in \text{FIRST}(S)$

$\# \in \text{LAST}(S) \Rightarrow \#$

$V \rightarrow T$

3. 构造算法优先关系矩阵。

$\rightarrow i$	i	+	b	*	a	#
$i \in \text{FIRSTVT}(T)$	\succ	\prec	\prec	\succ	\prec	\succ
+	\succ	\succ	\prec	\succ	\prec	\succ
b	\prec	\prec	\prec	\prec	\prec	
*	\succ	\succ		\succ		\succ

$\text{LASTVT}(T) \Rightarrow +$

$+ \in \text{FIRSTVT}(T)$

$$b \cup *$$

$$b \equiv *$$

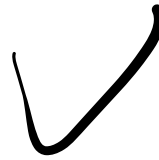
$$b \leq \text{FIRSTVT}(V)$$

$$\text{LASTVT}(V) \geq *$$

a	\Rightarrow	\Rightarrow		\Rightarrow		\Rightarrow
#	\Leftarrow	\Leftarrow	\Leftarrow		\Leftarrow	

4. 判断该文法是否为算符优先文法。

是，因为算符的优先关系唯一



$$A \rightarrow a \mid bS$$

G7S'J

5 → 5(11)

(2) $S \rightarrow SABC$

$$(3) S \rightarrow aBc$$

(4) $B \rightarrow bAc$

$$\text{Follow}(s') = \{ \# \}$$

$\text{Follow}(s) = \text{Follow}(s') \cup \text{Follow}(A)$
 $\quad \quad \quad \cup \{a\}$

Follow $\{B\} = \{C\} \Rightarrow \{a, c, \# \}$

$$\text{Follow}(A) = \{c\}$$

1. SLR 分析表 (Action 表和 GOTO 表)。

(5) $A \rightarrow a$

(6) $A \rightarrow bS$

(1) $S \rightarrow SaBc$

(2) $S \rightarrow aBc$

(3) $B \rightarrow bAc$

(4) $A \rightarrow a$

(5) $A \rightarrow bS$

10:

$$S = .SaBc$$
$$S = .aBc$$

I1: I0 + S

$$S = S.aBc$$

I2: I0 + a

$$S = a.Bc$$
$$B = .bAc$$

I3: I1 + a

$$S = S_{a.Bc}$$
$$B = .bAc$$

I4: I2 + B

$$S = aB.c$$

I5: I2 + b

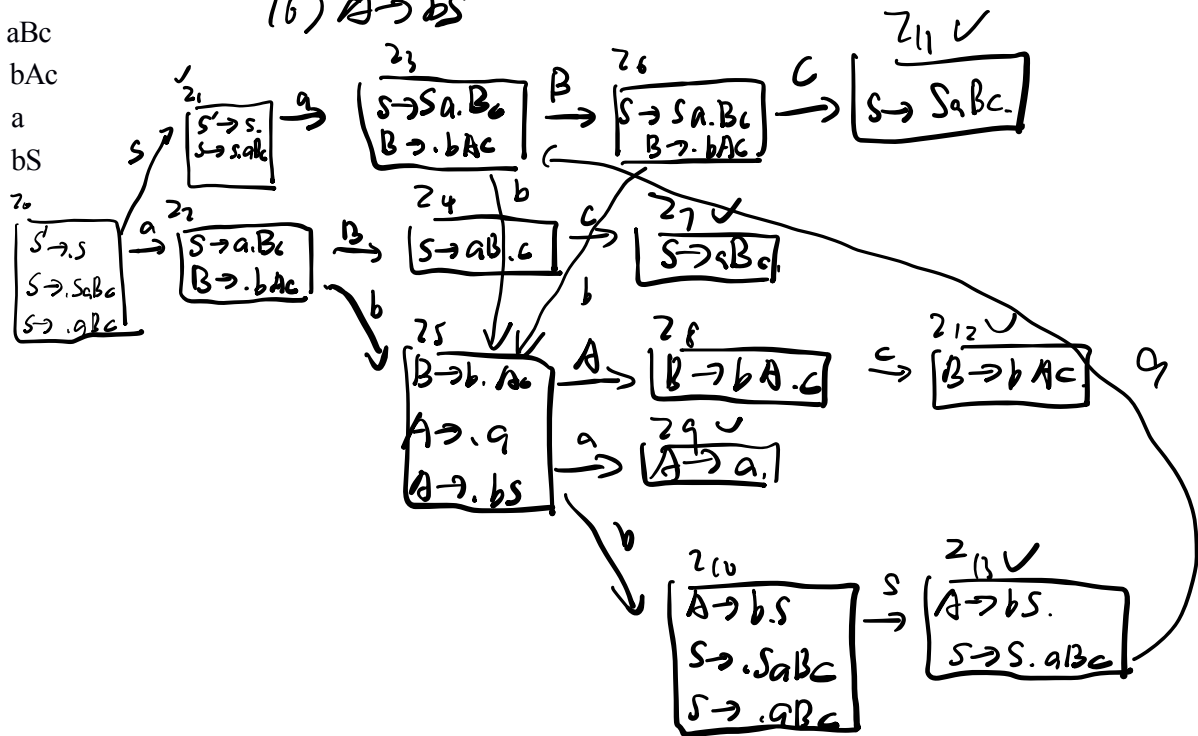
$$B = b.Ac$$
$$A = .a$$
$$A = .bS$$

I6: I3 + B

$$S = \text{SaB.c}$$

*I7: I3 + b

goto I5



	Action				Goto	
	a	b	c	#	S	A B
0	S2				1	
1	S3			accept		
2		S5				4
3		S5				6
4			S7			
5	S9	S0				8
6			S11			
7	r3		r3	r3		
8		S12				

I7: I4 + c
S = aBc.

I8: I5 + A
B = bA.c

I9: I5 + a
A = a.

I10: I5 + b
A = b.S
S = .SaBc
S = .aBc

I11: I6 + c
S = SaBc.

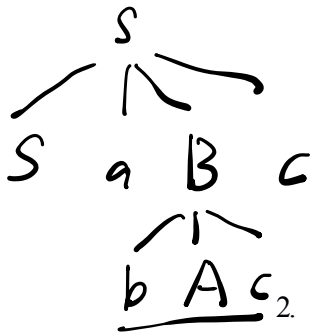
I12: I8 + c
B = bAc.

I13: I10 + S
A = bS.
S = S.aBc

*I14: I10 + a
goto I2

*I14: I13 + a
goto I3

	a	b	c	#	S	A	B
0	S2				1		
1	S3						
2		S5					4
3		S5					6
4			S7				
5	S9	S10				8	
6			S11				
7				r2			
8			S12				
9							
10	S2				13		
11				r1			
12				r3			
13	S3						



句柄

2. 识别句型 SabAcc 活前缀 SabAc 的有效项目集。

$Z_2 \{ B \rightarrow bAc \}$

五. 有如下程序段：(15 分)

```

program paser;
  var filename:string;
      curchar:char;
  procedure getsym;
    var symsize:integer;
    procedure getchar;
      var errorinfo:string;
    begin
      ... ----- (1)
    end;
  begin
    ...
    call getchar;
    ...
  end;
  procedure lexer;
    var linelength:integer;
  begin
    ...
    call getsym;----- (2)
    ...
  end;
begin
  ...
  call lexer;
  ...
end.

```

1. 请分别画出当编译到位置 (1) 和 (2) 时的符号表。

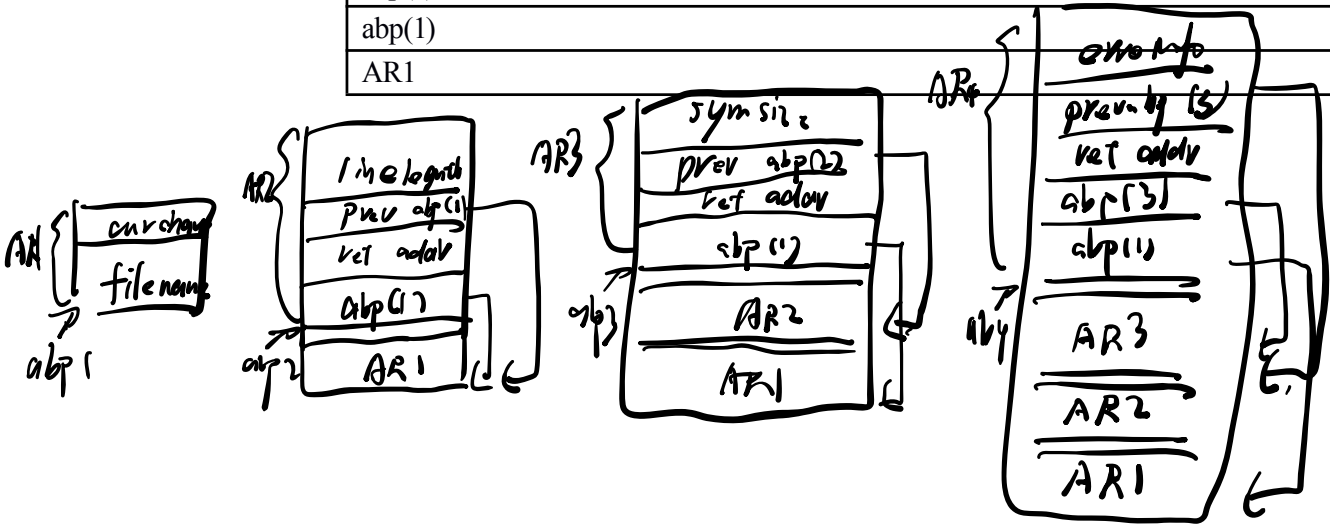
变量名	类型	维数	lev
filename	string	0	1
curchar	char	0	1
getsym	proc		1
symsize	integer	0	2
getchar	proc		2
errorinfo	string	0	3

变量名	类型	维数	
filename	string	0	1
curchar	char	0	1
getsym	proc		1
lexer	proc		1
linelength	integer		2

symsize	integer	0
getchar	proc	
errorinfo	string	0
lexer	proc	
linelength	integer	0

2. 运行到位置 (1) 时的运行栈。

errorinfo
prev abp(3)
ret addr(4)
abp(4)
abp(3)
abp(2)
abp(1)
symsize
prev abp(2)
ret addr(3)
abp(3)
abp(2)
abp(1)
linelength
prev abp(1)
ret addr(2)
abp(2)
abp(1)
AR1

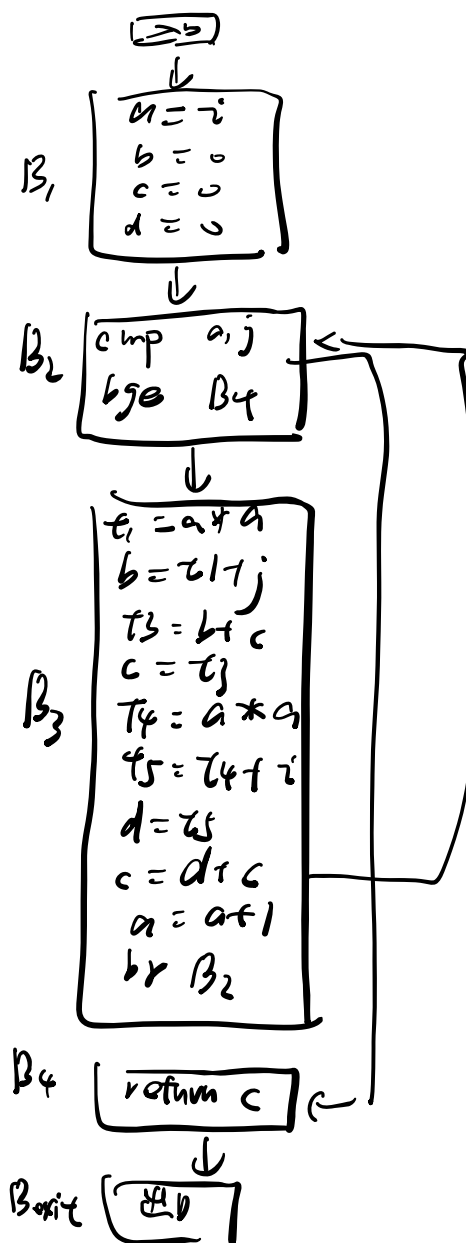


六. 有如下中间代码序列, 其中{a,b,c,d}为局部变量, {i,j}为形参, {t1,t2,t3,t4,t5,t6}为临时变量 (15分):

```

a = i
b = 0
c = 0
d = 0
_loop_begin:
    cmp a, j
    bge _loop_end // a >= j
    t1 = a * a
    b = t1 + j
    t3 = b + c
    c = t3
    t4 = a * a
    t5 = t4 + i
    d = t5
    c = d + c
    a = a + 1
    br _loop_begin // jmp _loop_begin
_loop_end:
    return c

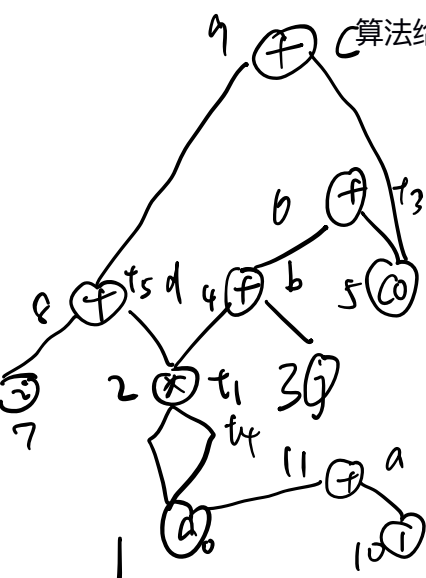
```



1. 对图中代码序列划分基本块, 构建流图。

2. 对循环体所在基本块利用 DAG 图做“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。



mode 00

A	1
t ₁	2
j	3
b	4
c	5
t ₃	6
t ₄	2
2	7
t ₅	8
d	8
1	10

11 a
7 c
8 t5
2 10 有 4 套
6 t3
4 b
2 t1

化, 并根据启发式

$$|a \ c \ t_5 \ t_3 \ b \ t_1|$$

排列

$$t_1 = a_0 * a_0$$

$$b = t_1 + j$$

$$t_3 = b + c$$

$$t_5 = t_1 + i$$

$$c = t_5 + t_3$$

$$a = a_0 +$$