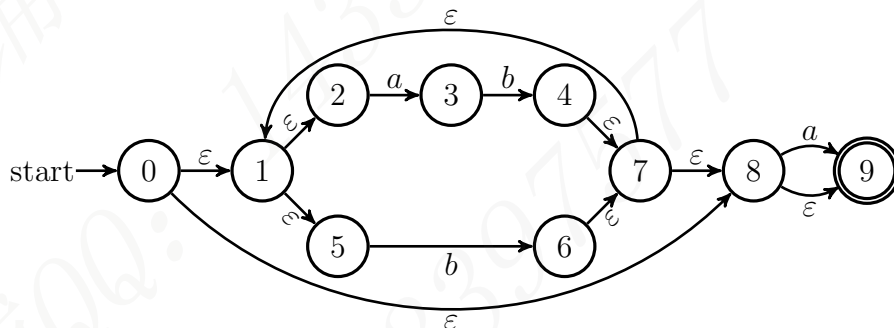


武汉大学计算机学院2007-2008学年第二学期 2005级《编译原理》考试试题

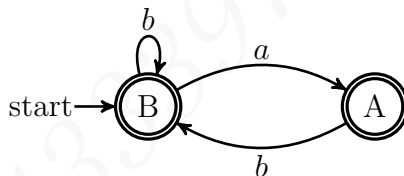
学号：_____ 姓名：_____ 成绩：_____

注意：请将答案全部写在答题纸上，并注明题号！

一、 设NFA N 的状态转换图如下所示： (20分, 5+5+5+5)



- (1) 试写出 N 接受输入串“abba”的过程；
- (2) 求正规表达式 r ，使得 $L(r) = L(N)$ ；
- (3) 用子集构造法求与 N 等价的DFA M ，要求有计算过程并画出 M 的状态转换图；
- (4) 设 N 对应的最小自动机如下所示，试用自然语言描述 N 所生成的语言。



二、 设文法 $G(S)$ 定义如下： (20分, 5+5+5+5)

$$\begin{aligned} S &\rightarrow aSb \mid B \\ B &\rightarrow Bb \mid \varepsilon \end{aligned}$$

- (1) 试写出语句“aabbbb”的一个最右推导并画出该语句对应的语法树；
- (2) 试描述文法 $G(S)$ 所生成的语言；
- (3) 试说明文法 $G(S)$ 不是LR(1)文法；
- (4) 设计一个与文法 $G(S)$ 等价的LR(1)文法。

三、 设文法 $G(S)$ 定义如下:

(15分, 5+5+5)

$$S \rightarrow aT$$

$$T \rightarrow +ST \mid \varepsilon$$

- (1) 试对文法 $G(S)$ 的每个非终结符求First集合和Follow集合;
- (2) 试构造文法 $G(S)$ 的LL(1)分析表, 从而说明该文法不是LL(1)文法;
- (3) 设计一个与 $G(S)$ 等价的LL(1)文法。

四、 设有函数定义和函数取值的表达式文法 $G(E)$ 定义如下:

(10分, 5+5)

$$E \rightarrow e \mapsto E \mid EE \mid (E) \mid e$$

其中: ‘ e ’, ‘ \mapsto ’, ‘(’和‘)’为终结符, “ $e \mapsto E$ ”为函数定义运算; “ EE ”为函数取值运算。

- (1) 试说明该文法是二义文法;
- (2) 设定义运算为右结合, 取值运算为左结合, 且取值运算的优先级高于定义运算, 设计一个与文法 $G(E)$ 等价的无二义文法, 使得其运算的优先级和结合次序与上述规定一致。

五、 设题四的拓广文法 $G(E')$ 定义如下:

(20分, 5+5+5+5)

$$E' \rightarrow E \quad (0)$$

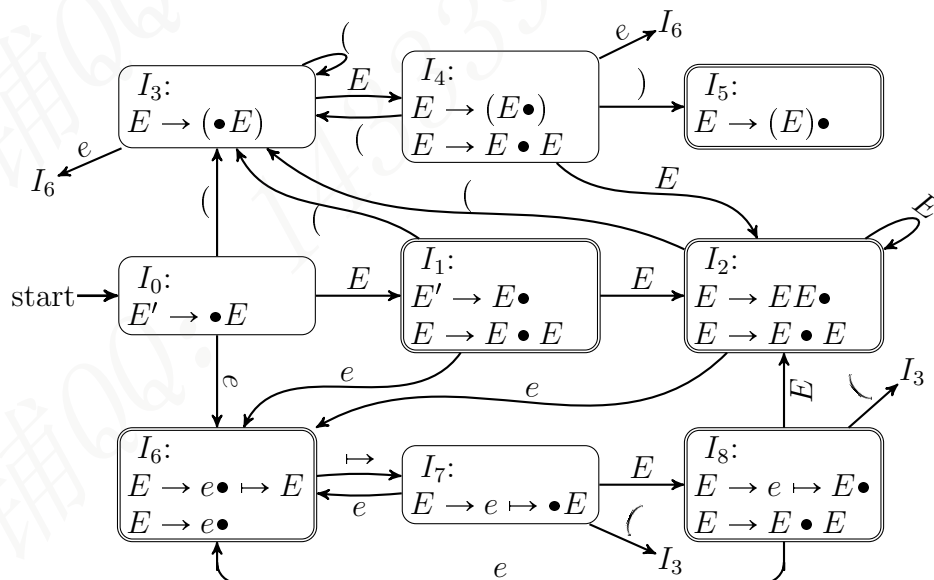
$$E \rightarrow e \mapsto E \quad (1)$$

$$\quad \quad \quad \mid EE \quad (2)$$

$$\quad \quad \quad \mid (E) \quad (3)$$

$$\quad \quad \quad \mid e \quad (4)$$

文法 $G(E')$ 的识别活前缀LR(0)项目自动机如下图所示(注意每个状态仅列出了核心项目):



- (1) 试问文法符号串“(E((Ee \mapsto E)”和“(e \mapsto E)E)”是否为文法G(E')的活前缀，如果是，请指出该活前缀对应的有效项目集；
- (2) 由于该文法是二义文法，因此其SLR分析表一定有移进/归约或归约/归约冲突，请指出哪些状态有怎样的冲突；
- (3) 试画出该文法的SLR分析表，使得其运算的优先级和结合次序与题四(2)规定一致；
- (4) 利用你的分析表写出输入表达式“e \mapsto ee”的分析过程。

六、 设短路法翻译控制流的语法制导定义如下所示： (10分, 5+5)

产生式	语义规则
$P \rightarrow S$	$S.next = newlabel(); \quad P.code = S.code \parallel label(S.next)$
$S \rightarrow exp$	$S.code = exp.code$
$S \rightarrow \text{if}(B) S_1$	$B.true = newlabel(); \quad B.false = S.next$ $S_1.next = S.next$ $S.code = B.code \parallel label(B.true) \parallel S_1.code$
$S \rightarrow \text{while}(B) S_1$	$begin = newlabel()$ $B.true = newlabel(); \quad B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \parallel B.code$ $\parallel label(B.true) \parallel S_1.code \parallel gen('goto' begin)$
$S \rightarrow S_1 S_2$	$S_1.next = newlabel(); \quad S_2.next = S.next$ $S.code = S_1.code \parallel label(S_1.next) \parallel S_2.code$
$S \rightarrow \{ S_1 \}$	$S.code = S_1.code; \quad S_1.next = S.next$
$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel(); \quad B_1.false = B.false$ $B_2.true = B.true; \quad B_2.false = B.false$ $B.code = B_1.code \parallel label(B_1.true) \parallel B_2.code$
$B \rightarrow E_1 \text{ rel } E_2$	$B.code = E_1.code \parallel E_2.code$ $\parallel gen('if', E_1.name \text{ rel.name } E_2.name 'goto' B.true)$ $\parallel gen('goto' B.false)$

其中：属性 $code$ 为对应非终结符的三地址代码链，属性 $B.true$ 和 $B.false$ 为对应布尔表达式的真假出口标号； $S.next$ 为紧随 S 之后的第一条三地址码所对应的标号；属性 $E.name$ 为保存表达式 E 结果的变量名或常量。函数 $gen()$ 生成三地址码；运算 \parallel 对两段三地址码链接； $newlabel()$ 产生一个新的标号； $label(L)$ 将为其后随的三地址码链的第一条语句加上标号 L 。

现对上述文法增加产生式 $S \rightarrow \text{break}$ 和 $S \rightarrow \text{continue}$ ，其语义同C语言一样，在 while 循环体中时， break 将跳出循环， continue 将结束本次循环；如果它们不被任何循环嵌套，将报错。为了实现语句 break 和 continue 的三地址码翻译，特对非终结符 S 引入两个新的继承属性 $break$ 和 $continue$ 。当 S 被 while 嵌套时， $S.break$ 对应于紧随循环体之后的第一条三地址码的标号， $S.continue$ 应于嵌套 S 的最内层循环语句三地址代码链的第一条语句的标号；当 S 不被 while 嵌套时， $S.break$ 和 $S.continue$ 均为 \emptyset 。

- (1) 试按照上述语义对上述每一条关于 S 的产生式和产生式 $P \rightarrow S$ 写属性 $S.break$ 和 $S.continue$ 的语法制导定义, 为产生式 $S \rightarrow \text{break}$ 和 $S \rightarrow \text{continue}$ 写属性 $S.code$ 的语法制导定义。
- (2) 按照上述语法制导定义写出下述语句对应的三地址码:

```
while (x > 0 && x < 100) {  
    x := x + 1  
    if (x = 10) continue  
    x := x * 2  
    if (x = 100) break  
}
```

七、设有如下C语言程序:

(5分)

```
#include <stdio.h>  
  
void f(int, int);  
int main()  
{  
    f(5, 1);  
    return 0;  
}  
  
void f ( int parameter, int result )  
{  
    int *ptr = &parameter;  
    printf ("parameter = %d,\tresult = %d\n", parameter, result);  
    if (parameter == 1) return;  
    result = result * parameter--;  
    *(ptr - 1) = *(ptr - 1) - 5;  
    return;  
}
```

在Intel x86/Linux下用GCC编译没有任何警告, 运行编译后的程序输出如下结果并正常结束:

```
parameter = 5,  result = 1  
parameter = 4,  result = 5  
parameter = 3,  result = 20  
parameter = 2,  result = 60  
parameter = 1,  result = 120
```

试问该程序为什么递归调用了函数f, 最后输出5的阶乘。