

武汉大学计算机学院2007-2008学年第二学期  
2005级《编译原理》参考答案

一、 (1)



(2)  $(ab|b)^*(a|\varepsilon)$

(3)

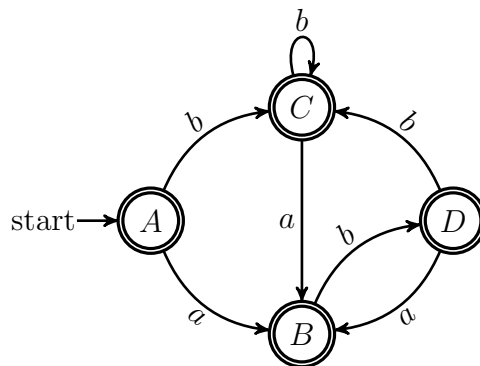
$$A = \{0, 1, 2, 5, 8, 9\}$$

$$B = \{3, 9\}$$

$$C = \{1, 2, 5, 6, 7, 8, 9\}$$

$$D = \{1, 2, 4, 5, 7, 8, 9\}$$

状态转换图为：

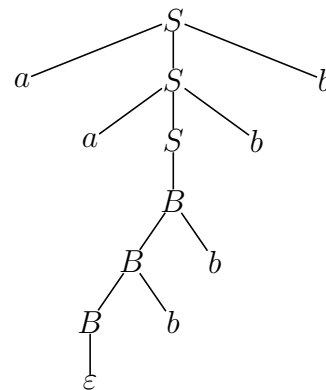


(4) 没有 $aa$ 子串；或没有连续的 $a$ 。

二、 (1) 最右推导如下：

$$\begin{aligned} S &\xRightarrow{rm} aSb \\ &\xRightarrow{rm} aaSbb \\ &\xRightarrow{rm} aaBbb \\ &\xRightarrow{rm} aaBbbb \\ &\xRightarrow{rm} aaBbbbb \\ &\xRightarrow{rm} aabbbb \end{aligned}$$

语法树：



$$(2) \{ a^m b^n \mid m, n \in \mathbb{N} \wedge m \leq n \}$$

- (3) 由上最右推导得知，LR分析器识别语句 $a^m b^p b^m (p \geq 0)$ 的步骤应该是：  
在移进 $n$ 个 $a$ 之后，先把前 $p$ 个多余的 $b$ 归约为 $B$ ，再将 $B$ 归约为 $S$ 得到活前缀 $a^n S$ ，最后将剩余的同 $a$ 的次数平衡的 $m$ 个 $b$ 用产生式  $S \rightarrow aSb$  逐个归约。由于多余的 $b$ 可以任意多，且LR分析从左到右的扫描机制，及只能向前查看固定次数的符号，因此分析器无法知道有多少个 $b$ 是多余的，从而无法解决何时停止将多余的 $b$ 归约为 $B$ 。故不是LR(k)文法。  
或者简答为：LR分析器由于无法知道有多少个 $b$ ，因此不能判断多少个 $b$ 归约为 $B$ ，多少个 $b$ 用于平衡 $a$ 。

- (4) 修改 $a$ 和 $b$ 匹配的机制即可，让在前面出现的 $b$ 先与 $a$ 匹配：

$$S \rightarrow TB$$

$$T \rightarrow aTb \mid \varepsilon$$

$$B \rightarrow Bb \mid \varepsilon$$

- 三、 (1)  $\text{First}(S) = \{ a \}; \text{First}(T) = \{ +, \varepsilon \}$   
 $\text{Follow}(S) = \{ +, \$ \}; \text{Follow}(T) = \{ +, \$ \}$

- (2)

|     | +   | $a$                | $\$$                        |
|-----|---|--------------------|-----------------------------|
| $S$ |   | $S \rightarrow aT$ |                             |
| $T$ | $T \rightarrow +ST \quad T \rightarrow \varepsilon$ |                    | $T \rightarrow \varepsilon$ |

这样面对栈顶符号 $T$ 和输入单词 $+$ 可选两个不同的产生展开，故不是LL(1)文法。

- (3) 实际上上述文法是二义文法，生成的语言是仅有加法运算的表达式，其LL(1)文法如下：

$$S \rightarrow aT$$

$$T \rightarrow +aT \mid \varepsilon$$

四、 (1) 面对输入“ $e \mapsto ee$ ”有两个不同的最左推导。

推导1:

$$\begin{aligned}
 E &\xRightarrow{lm} EE \\
 &\xRightarrow{lm} e \mapsto EE \\
 &\xRightarrow{lm} e \mapsto eE \\
 &\xRightarrow{lm} e \mapsto ee
 \end{aligned}$$

推导2:

$$\begin{aligned}
 E &\xRightarrow{lm} e \mapsto E \\
 &\xRightarrow{lm} e \mapsto EE \\
 &\xRightarrow{lm} e \mapsto eE \\
 &\xRightarrow{lm} e \mapsto ee
 \end{aligned}$$

(2)

$$\begin{aligned}
 E &\rightarrow e \mapsto E \mid T \\
 T &\rightarrow TF \mid F \\
 F &\rightarrow e \mid (E)
 \end{aligned}$$

五、 (1) 识别活前缀的自动机在吃进  $(E((Ee \mapsto E$  之后到达状态 $I_8$ ，因此它是活前缀，其对应的有效项目集即是 $I_8$ 所对应的项目集：

$$\begin{aligned}
 \overline{\{E \rightarrow e \mapsto E\bullet, E \rightarrow E\bullet E\}} &= \{E \rightarrow e \mapsto E\bullet, E \rightarrow E\bullet E, \\
 &E \rightarrow \bullet e \mapsto E, E \rightarrow \bullet EE, E \rightarrow \bullet(E), E \rightarrow \bullet e\}
 \end{aligned}$$

识别活前缀的自动机在吃进  $((e \mapsto E)$  之后到达状态 $I_4$ ，不能再接受任何非终结符，因此  $((e \mapsto E)E$  不是活前缀。

(2) 状态 $I_2$ 和 $I_8$ 有移进/归约冲突。

(3)

| 状态 | action    |    |    |     |      | goto |
|----|-----------|----|----|-----|------|------|
|    | $\mapsto$ | (  | )  | $e$ | $\$$ | $E$  |
| 0  |           | s3 |    | s6  |      | 1    |
| 1  |           | s3 |    | s6  | acc  | 2    |
| 2  |           | r2 | r2 | r2  | r2   | 2    |
| 3  |           | s3 |    | s6  |      | 4    |
| 4  |           | s3 | s5 | s6  |      | 2    |
| 5  |           | r3 | r3 | r3  | r3   |      |
| 6  | s7        | r4 | r4 | r4  | r4   |      |
| 7  |           | s3 |    | s6  |      | 8    |
| 8  |           | s3 | r1 | s6  | r1   | 2    |

(4)

| 剩余串              | 分析栈                 | 分析动作                               |
|------------------|---------------------|------------------------------------|
| $e \mapsto ee\$$ | 0                   | shift                              |
| $\mapsto ee\$$   | 0e6                 | shift                              |
| $ee\$$           | 0e6 $\mapsto 7$     | shift                              |
| $e\$$            | 0e6 $\mapsto 7e6$   | reduce $E \rightarrow e$           |
| $e\$$            | 0e6 $\mapsto 7E8$   | shift                              |
| $\$$             | 0e6 $\mapsto 7E8e6$ | reduce $E \rightarrow e$           |
| $\$$             | 0e6 $\mapsto 7E8E2$ | reduce $E \rightarrow EE$          |
| $\$$             | 0e6 $\mapsto 7E8$   | reduce $E \rightarrow e \mapsto E$ |
| $\$$             | 0S1                 | 分析成功                               |

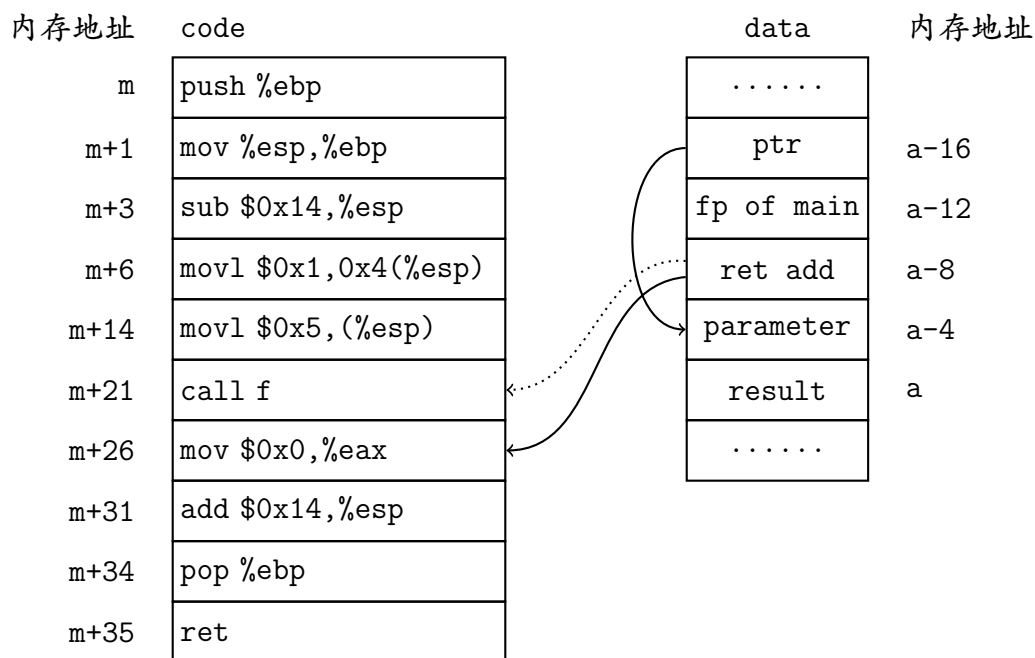
六、 (1)

| 产生式                                 | 语义规则   |
|-------------------------------------|--|
| $P \rightarrow S$                   | $S.break = \emptyset; \quad S.continue = \emptyset$  |
| $S \rightarrow \text{if}(B) S_1$    | $S_1.break = S.break; \quad S_1.continue = S.continue$   |
| $S \rightarrow \text{while}(B) S_1$ | $begin = newlabel()$<br>$S_1.break = S.next; \quad S_1.continue = begin$   |
| $S \rightarrow S_1 S_2$             | $S_1.break = S.break; \quad S_1.continue = S.continue$<br>$S_2.break = S.break; \quad S_2.continue = S.continue$                           |
| $S \rightarrow \{ S_1 \}$           | $S_1.break = S.break; \quad S_1.continue = S.continue$   |
| $S \rightarrow \text{break}$        | <b>if</b> ( $S.break == \emptyset$ ) <b>then</b><br>$error(\text{"break not within while"})$<br><b>else</b> $gen('goto' S.break)$          |
| $S \rightarrow \text{continue}$     | <b>if</b> ( $S.continue == \emptyset$ ) <b>then</b><br>$error(\text{"continue not within while"})$<br><b>else</b> $gen('goto' S.continue)$ |

(2) 对应的三地址码:

|  |   |
|--|---|
| <pre> 14: if (x &gt; 0) goto 10       goto 16 10: if (x &lt; 100) goto 15       goto 16 15: t0 := x + 1       x := t0       if (x = 10) goto 11       goto 12 </pre> | <pre> 11: goto 14 12: t1 := x * 2       x := t1       if (x = 100) goto 13       goto 14 13: goto 16       goto 14 16: </pre> |
|--|---|

七、 程序在调用f()后，内存的格局如下图所示：



语句“\*(ptr - 1) = \*(ptr - 1) - 5;”把内存“ret add”的值修改成“call f”指令所在的地址，这样函数f()返回后将继续“call f”直到从“if ( parameter == 1)”为真出口返回。

或简答：语句“\*(ptr - 1) = \*(ptr - 1) - 5;”修改了返回地址。