考试科目名称 编译原理; A 卷

考试方式: 闭卷　考试日期：2021 年 01 月 17 日　教师：魏恒峰

院系（专业）：软件学院 (软件工程)　　年级：2018 级

姓名：_____　学号：_____　成绩：_____

题号	一	二	三	四	五	六				
分数										

注意事项:

- 诚信考试, 不得作弊。
- 若对题意有疑问, 请及时提出。
- **题目不是按照难度排列的, 请注意合理分配时间**。
- 为了避免连锁错误造成的影响, 请尽量给出关键步骤。
- 题目中的分值仅表明各小题之间的相对权重, 不代表实际分数。
- 提示: 试题后的提示仅作提示之用, 不排除其它解法。
- **字迹要尽可能工整, 解题过程要尽可能清晰**。

题目 1 (正则表达式与自动机 (10 = 3 : 4 : 3))

考虑正则表达式 $r = \epsilon|(a^*b)$。

(1) 请使用 Thompson 构造法构造等价的 NFA;

(2) 请使用子集构造法构造等价的 DFA (注: 请标明状态之间的对应关系);

(3) 请将上一步构造的 DFA 最小化。

题目 2 (LL 语法分析 ($10 = 2:4:3:1$))

考虑文法 G :

$$S \to AB \tag{1}$$

$$A \to Aaa|a \tag{2}$$

$$B \to Bbb|b \tag{3}$$

(1) 请消除文法 G 中的左递归 (记新文法为 G');

(2) 请为文法 G' 计算必要的 First 集合与 Follow 集合;

(3) 请为文法 G' 构造预测分析表;

(4) 请问文法 G' 是否是 $LL(1)$ 文法, 并说明理由。

题目 3 (LR 语法分析 ($10 = 2:3:2:3$ 修改为 $10 = 3:4:3$))

考虑文法 G (已作增广处理):

$$S' \to S \tag{0}$$
$$S \to V = E \tag{1}$$
$$S \to E \tag{2}$$
$$E \to V \tag{3}$$
$$V \to x \tag{4}$$
$$V \to *E \tag{5}$$

(1) 请为文法 G 计算必要的 First 集合与 Follow 集合;

(2) 请为文法 G 构造 $LR(1)$ 自动机;

注意: 为了尽量统一状态编号, 便于批改, 当计算 closure 时, 请按照文法编号大小顺序加入新项。当计算 goto(I, X) 时, 请按照 I 中项的出现顺序依次考虑可能的转移符号 X。

要求: 请说明归约的设置条件。

(3) 请为文法 G 设计 $LR(1)$ 分析表; 该文法是 $LR(1)$ 文法吗? 请说明理由。如果该文法是 $LR(1)$ 文法, 请给出识别输入串 $*x = **x$ 时自动机所经历的状态 (编号)。

(4) **[删除]** 为该文法设计 $LALR(1)$ 分析表; 该文法是 $LALR(1)$ 文法吗? 请说明理由。

题目 4 (语法制导定义与翻译 (10 = 6 : 4))

考虑如下文法 G,

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

请设计语法制导的**翻译方案**, 完成下列任务。你需要自行定义合适的属性。

(1) 计算每个 a 的嵌套深度。例如, 在句子 $(a,(a,a))$ 中, 每个 a 的嵌套深度分别为 $1,2,2$。

(2) 计算每个 a 的位置。例如, 在句子 $(a,(a,(a,a),(a)))$ 中, 每个 a 的位置分别为 $2,5,8,10,14$。

题目 5 (中间代码生成 (10 = 3 : 3 : 4 修改为 10 = 4 : 1 : 5))

考虑循环语句

$$\textbf{do } S \textbf{ while } B$$

(1) 请基于控制流语句与布尔表达式语法制导定义 (见图2与图3) 为 **do**...**while** 语句添加语义规则;

(2) 请基于布尔表达式与控制流语句回填翻译方案 (见图4与图5) 为 **do**...**while** 语句设计回填方案。

(3) 请为以下代码片段生成中间代码 (注: 请自行选择是否使用回填方案)

do

  if $i == 2021$

    print *"Happy New Year"*

  $i = i + 1$

while $(i > 2000 \;\&\&\; i < 3000)$

(考试后修正: $i = i + 1$ 放在 if 外; 不影响解题)

要求: 请使用图示 (如注释语法树等) 展示产生式与相应规则的使用情况。

题目 6 (语法分析算法设计 (附加题: 5 分))

如果某上下文无关文法的每个产生式都是下列形式之一, 则该文法是 CNF (Chomsky Normal Form) 文法:

- $S \to \epsilon$, 其中 S 必须是起始符号;
- $A \to BC$, 其中 B, C 均为非终结符;
- $A \to a$, 其中 a 为终结符。

请使用动态规划思想设计语法分析算法, 判定输入串 w 是否符合某 CNF 文法 G。

(注: 你设计的算法**不要求**从左到右仅扫描一遍输入串; 可以假定输入串已经全部读入。)

(提示: 考虑输入串的任一子串以及任一非终结符。)

图1与 $w =$ she eats a fish with a fork 作为例子供参考。

$$
\begin{aligned}
\text{S} &\longrightarrow \text{NP VP} \\
\text{VP} &\longrightarrow \text{VP PP} \\
\text{VP} &\longrightarrow \text{V NP} \\
\text{VP} &\longrightarrow \text{eats} \\
\text{PP} &\longrightarrow \text{P NP} \\
\text{NP} &\longrightarrow \text{Det N} \\
\text{NP} &\longrightarrow \text{she} \\
\text{V} &\longrightarrow \text{eats} \\
\text{P} &\longrightarrow \text{with} \\
\text{N} &\longrightarrow \text{fish} \\
\text{N} &\longrightarrow \text{fork} \\
\text{Det} &\longrightarrow \text{a}
\end{aligned}
$$

图 1: CNF 文法示例 G

产生式	语义规则
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$
$S \rightarrow$ **assign**	$S.code =$ **assign**$.code$
$S \rightarrow$ **if** (B) S_1	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$S \rightarrow$ **if** (B) S_1 **else** S_2	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ S.next)$ $\mid\mid label(B.false) \mid\mid S_2.code$
$S \rightarrow$ **while** (B) S_1	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \mid\mid B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ begin)$
$S \rightarrow S_1\ S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code$

图 2: 控制流语句的语法制导定义

产生式	语义规则
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \,\&\&\, B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow !\, B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1$ **rel** E_2	$B.code = E_1.code \mid\mid E_2.code$ $\mid\mid gen('\texttt{if}'\ E_1.addr$ **rel**$.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\mid\mid gen('\texttt{goto}'\ B.false)$
$B \rightarrow$ **true**	$B.code = gen('\texttt{goto}'\ B.true)$
$B \rightarrow$ **false**	$B.code = gen('\texttt{goto}'\ B.false)$

图 3: 布尔表达式的语法制导定义

1)	$B \rightarrow B_1$ || M B_2	{ $backpatch(B_1.falselist, M.instr)$; $B.truelist = merge(B_1.truelist, B_2.truelist)$; $B.falselist = B_2.falselist$; }
2)	$B \rightarrow B_1$ && M B_2	{ $backpatch(B_1.truelist, M.instr)$; $B.truelist = B_2.truelist$; $B.falselist = merge(B_1.falselist, B_2.falselist)$; }
3)	$B \rightarrow$! B_1	{ $B.truelist = B_1.falselist$; $B.falselist = B_1.truelist$; }
4)	$B \rightarrow$ (B_1)	{ $B.truelist = B_1.truelist$; $B.falselist = B_1.falselist$; }
5)	$B \rightarrow E_1$ **rel** E_2	{ $B.truelist = makelist(nextinstr)$; $B.falselist = makelist(nextinstr + 1)$; $gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}\ \_')$; $gen('\texttt{goto}\ \_')$; }
6)	$B \rightarrow$ **true**	{ $B.truelist = makelist(nextinstr)$; $gen('\texttt{goto}\ \_')$; }
7)	$B \rightarrow$ **false**	{ $B.falselist = makelist(nextinstr)$; $gen('\texttt{goto}\ \_')$; }
8)	$M \rightarrow \epsilon$	{ $M.instr = nextinstr$; }

图 4: 布尔表达式的回填翻译方案

1)	$S \rightarrow$ **if** (B) M S_1	{ $backpatch(B.truelist,\ M.instr)$; $S.nextlist = merge(B.falselist,\ S_1.nextlist)$; }
2)	$S \rightarrow$ **if** (B) M_1 S_1 N **else** M_2 S_2	{ $backpatch(B.truelist,\ M_1.instr)$; $backpatch(B.falselist,\ M_2.instr)$; $temp = merge(S_1.nextlist,\ N.nextlist)$; $S.nextlist = merge(temp,\ S_2.nextlist)$; }
3)	$S \rightarrow$ **while** M_1 (B) M_2 S_1	{ $backpatch(S_1.nextlist,\ M_1.instr)$; $backpatch(B.truelist,\ M_2.instr)$; $S.nextlist = B.falselist$; $gen('\texttt{goto}'\ M_1.instr)$; }
4)	$S \rightarrow$ { L }	{ $S.nextlist = L.nextlist$; }
5)	$S \rightarrow A$;	{ $S.nextlist =$ **null**; }
6)	$M \rightarrow \epsilon$	{ $M.instr = nextinstr$; }
7)	$N \rightarrow \epsilon$	{ $N.nextlist = makelist(nextinstr)$; $gen('\texttt{goto}\ \_')$; }
8)	$L \rightarrow L_1$ M S	{ $backpatch(L_1.nextlist,\ M.instr)$; $L.nextlist = S.nextlist$; }
9)	$L \rightarrow S$	{ $L.nextlist = S.nextlist$; }

图 5: 控制流语句的回填翻译方案