

《编译原理》第三次书面作业

截止日期：2024 年 11 月 19 日

若发现问题，或有任何想法（改进题目、调整任务量等等），请及时联系助教

Q1. 以下是某简单语言的一段代码。其语法基本与 C 语言一致，但支持嵌套的函数声明，且允许全局语句（类似 Python）。

```
1  int a0, b0, a2;
2  void foo() {
3      int a1, b1;
4      void bar() {
5          int a2;
6          a2 = a1 + b1;
7          if (a0 != b0) baz();
8      }
9      a1 = a0 - b0;
10     b1 = a0 + b0;
11     if (a1 < b1) bar();
12 }
13 void baz() {
14     int a3;
15     a3 = a0 * b0;
16     if (a2 != a3) foo();
17 }
18 a0 = 1;
19 b0 = 2;
20 a2 = a0 - b0;
21 baz();
```

1. 假设该语言编译器采用一个全局的单符号表栈结构。试指出：在分析至语句 11 和 16 时，当前开作用域分别有几个？各包含哪些符号？在分析至语句 16 时，所访问的 `a2` 是在哪行语句声明的？
2. 假设该语言编译器采用多符号表的组织和管理方式，即每个静态作用域均对应一个符号表；且该语言编译器采用多遍扫描机制，在静态语义检查之前每个作用域中的所有表项均已生成。试指出：在分析至语句 11 和 16 时，当前开作用域分别有几个？各包含哪些符号？

参考答案：

1. 在分析至语句 11 时，开作用域：全局作用域（含符号 `a0, b0, a2, foo`），`foo` 作用域（含符号 `a1, b1, bar`）；在分析至语句 16 时，开作用域：全局作用域（含符号 `a0, b0, a2, foo, baz`），`baz` 作用域（含符号 `a3`）。在分析至语句 16 时，所访问的 `a2` 是在第 1 行语句声明的。 **本题没有考虑 `if` 语句的作用域，下同，考试会避免此类歧义出现，感谢王玮康同学指出。**
2. 在分析至语句 11 时，开作用域：全局作用域（含符号 `a0, b0, a2, foo, baz`），`foo` 作用域（含符号 `a1, b1, bar`）；在分析至语句 16 时，开作用域：全局作用域（含符号 `a0, b0, a2, foo, baz`），`baz` 作用域（含符号 `a3`）。

Q2. 现有文法 $G[S]$ 的翻译模式：

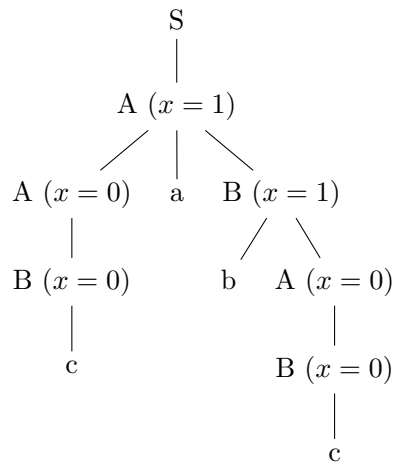
$$\begin{aligned} S &\rightarrow A \{ \text{print}(A.x) \} \\ A &\rightarrow A_1 a B \{ A.x := A_1.x + B.x \} \\ A &\rightarrow B \{ A.x := B.x \} \\ B &\rightarrow b A \{ B.x := A.x + 1 \} \\ B &\rightarrow c \{ B.x = 0 \} \end{aligned}$$

1. 对于输入串 $cabc$ ，最终会打印出什么？请画出带标注的语法分析树。
2. 该文法含有左递归，无法应用 LL(1) 分析。请消除文法中的左递归（新引入的非终结符用 R 表示），并给出消除后的翻译模式。
3. 针对消除左递归后的翻译模式构造递归下降分析程序，下面给出了相关全局变量和函数的声明，其中 `matchToken(char)` 的定义与课件一致。请写出 `parseR(int)` 的定义。

```
1 static char lookahead;
2 void parseS();
3 int parseA();
4 int parseB();
5 int parseR(int R_i);
6 void matchToken(char expected);
```

参考答案：

1. 输出 1。语法分析树如下所示。



2. 如下所示。

$$\begin{aligned} S &\rightarrow A \{ \text{print}(A.x) \} \\ A &\rightarrow B \{ R.i := B.x \} R \{ A.x := R.s \} \\ R &\rightarrow aB \{ R_1.i := R.i + B.x \} R_1 \{ R.s := R_1.s \} \\ R &\rightarrow \epsilon \{ R.s := R.i \} \\ B &\rightarrow bA \{ B.x := A.x + 1 \} \\ B &\rightarrow c \{ B.x = 0 \} \end{aligned}$$

3. 如下所示。（不必拘泥于语法细节）感谢李梓晟同学指出变换后的文法仍不是 LL(1) 文法。下面的答案应用了额外规则：遇到冲突时，优先匹配非空生成式。

```

1 int parseR(int R_i) {
2     int R1_i;
3     switch (lookahead) {
4         case 'a':
5             matchToken('a');
6             R1_i = R_i + parseB();
7             return parseR(R1_i);
8         case '#':
9             return R_i;
10        default:
11            printf("syntax error\n");
12            exit(1);
13    }
14 }

```

Q3. 现有文法 $G[S]$ 的翻译模式（注意；和，都是该语言的终结符）：

```

 $S \rightarrow D \{ \text{print}(D.\text{width}) \}$ 
 $D \rightarrow D_1; T \{ L.\text{type} := T.\text{type}; L.\text{offset} := D_1.\text{width}; L.\text{width} := T.\text{width} \} L$ 
 $\quad \{ D.\text{width} := D_1.\text{width} + L.\text{num} * T.\text{width} \}$ 
 $D \rightarrow T \{ L.\text{type} := T.\text{type}; L.\text{offset} := 0; L.\text{width} := T.\text{width} \} L$ 
 $\quad \{ D.\text{width} := L.\text{num} * T.\text{width} \}$ 
 $T \rightarrow \underline{\text{int}} \{ T.\text{type} := 'int'; T.\text{width} := 4 \}$ 
 $T \rightarrow \underline{\text{real}} \{ T.\text{type} := 'real'; T.\text{width} := 8 \}$ 
 $L \rightarrow \{ L_1.\text{type} := L.\text{type}; L_1.\text{offset} := L.\text{offset}; L_1.\text{width} := L.\text{width} \} L_1, \underline{id}$ 
 $\quad \{ \text{foo}(\text{id.name}, L.\text{type}, L.\text{offset} + L_1.\text{num} * L.\text{width}); L.\text{num} := L_1.\text{num} + 1 \}$ 
 $L \rightarrow \underline{id} \{ \text{foo}(\text{id.name}, L.\text{type}, L.\text{offset}); L.\text{num} := 1 \}$ 

```

1. 变换翻译模式，使嵌在产生式中间的语义动作集中仅含复写规则，且在自底向上的语法分析过程中，文法符号的所有继承属性均可以通过归约前已出现在分析栈中的确定的综合属性进行访问。
2. 在自底向上分析过程中，语法栈为 v ，栈顶位置为 top ，可以用 $v[\text{top}+i].\text{width}$ 访问 $\text{top}+i$ 位置的非终结符的 width 属性。写出在按每个产生式归约时语义计算的一个代码片断。不用考虑对 top 的维护。
3. 对于输入串 $\underline{\text{int}} \underline{\text{id}}, \underline{\text{id}}$ ，最终会打印出什么？（虽然可以靠语义猜出来，但建议还是用 LR 分析过程完整算一遍）

参考答案：

1. 需要变换的只有 D 的两个产生式，因为在其语义动作中， L 的继承属性无法确定地从栈上得

到。变换后的翻译模式如下所示。

$$\begin{aligned}
S &\rightarrow D \{ \text{print}(D.\text{width}) \} \\
D &\rightarrow D_1; T \{ L.\text{type} := T.\text{type}; L.\text{offset} := D_1.\text{width}; L.\text{width} := T.\text{width} \} L \\
&\quad \{ D.\text{width} := D_1.\text{width} + L.\text{num} * T.\text{width} \} \\
D &\rightarrow MNT \{ L.\text{type} := T.\text{type}; L.\text{offset} := M.\text{width}; L.\text{width} := T.\text{width} \} L \\
&\quad \{ D.\text{width} := L.\text{num} * T.\text{width} \} \\
T &\rightarrow \underline{int} \{ T.\text{type} := 'int'; T.\text{width} := 4 \} \\
T &\rightarrow \underline{real} \{ T.\text{type} := 'real'; T.\text{width} := 8 \} \\
L &\rightarrow \{ L_1.\text{type} := L.\text{type}; L_1.\text{offset} := L.\text{offset}; L_1.\text{width} := L.\text{width} \} L_1, \underline{id} \\
&\quad \{ \text{foo}(\text{id}.\text{name}, L.\text{type}, L.\text{offset} + L_1.\text{num} * L.\text{width}); L.\text{num} := L_1.\text{num} + 1 \} \\
L &\rightarrow \underline{id} \{ \text{foo}(\text{id}.\text{name}, L.\text{type}, L.\text{offset}); L.\text{num} := 1 \} \\
M &\rightarrow \epsilon \{ M.\text{width} := 0 \} \\
N &\rightarrow \epsilon \{ \}
\end{aligned}$$

2. 如下所示。

$$\begin{aligned}
S &\rightarrow D \{ \text{print}(v[\text{top}].\text{width}) \} \\
D &\rightarrow D_1; TL \{ v[\text{top-3}].\text{width} := v[\text{top-3}].\text{width} + v[\text{top}].\text{num} * v[\text{top-1}].\text{width} \} \\
D &\rightarrow MNTL \{ v[\text{top-3}].\text{width} := v[\text{top}].\text{num} * v[\text{top-1}].\text{width} \} \\
T &\rightarrow \underline{int} \{ v[\text{top}].\text{type} := 'int'; v[\text{top}].\text{width} := 4 \} \\
T &\rightarrow \underline{real} \{ v[\text{top}].\text{type} := 'real'; v[\text{top}].\text{width} := 8 \} \\
L &\rightarrow L_1, \underline{id} \\
&\quad \{ \text{foo}(v[\text{top}].\text{name}, v[\text{top-3}].\text{type}, v[\text{top-5}].\text{width} + v[\text{top-2}].\text{num} * v[\text{top-3}].\text{width}); \\
&\quad v[\text{top-2}].\text{num} := v[\text{top-2}].\text{num} + 1 \} \\
L &\rightarrow \underline{id} \{ \text{foo}(v[\text{top}].\text{name}, v[\text{top-1}].\text{type}, v[\text{top-3}].\text{width}); v[\text{top}].\text{num} := 1 \} \\
M &\rightarrow \epsilon \{ v[\text{top+1}].\text{width} := 0 \} \\
N &\rightarrow \epsilon \{ \}
\end{aligned}$$

解析：对于产生式 $L \rightarrow L_1, \underline{id}$ ，归约时语法栈如下所示。此时 $L.\text{type}$ 继承自 $T.\text{type}$ ， $L.\text{offset}$ 继承自 $D.\text{width}$ 或 $M.\text{width}$ ； $L.\text{width}$ 继承自 $T.\text{width}$ 。

...	D 或 M	; 或 N	T	L_1	,	\underline{id}
-----	-----------	---------	-----	-------	---	------------------

对于产生式 $L \rightarrow \underline{id}$ ，归约时语法栈如下所示。继承关系与上面相同，只是栈顶符号排列不同。

...	D 或 M	; 或 N	T	\underline{id}
-----	-----------	---------	-----	------------------

3. 输出 8。