

2021 年第 4 次作业：语法制导翻译

Hollow Man

1

(50 分) 写一个 SDD，完成下面的题目：

❖ 在C语言中，自增运算符只能作用于“左值”（如变量名），而 $3++$ 和 $(id + id)++$ 这样的表达式在编译时都会得到如下的错误提示：

invalid lvalue in increment

现有如下简化的C语言表达式文法：

$E \rightarrow E + E \mid (E) \mid E ++ \mid id \mid number$

写出一个语法制导定义或翻译方案，它检查++的运算对象是否合法。

给非终结符 E 一个综合属性 v，其值可取 lvalue 或 rvalue，分别表示 E 是左值表达式和右值表达式，那么语法制导定义如下（无输出则表示无错）：

	产生式	语法规则
1)	$E \rightarrow E1 + E2$	$E.v = rvalue$
2)	$E \rightarrow (E1)$	$E.v = E1.v$
3)	$E \rightarrow E1 ++$	if $E1.v == rvalue$ then printf("invalid lvalue in increment"); $E.v := rvalue$
4)	$E \rightarrow id$	$E.v := lvalue$
5)	$E \rightarrow num$	$E.v := rvalue$

2

(50 分) 以作业二中的后缀表达式文法为基础：

$S \rightarrow S S + \mid S S * \mid id$

设计一个语法制导定义（SDD），将每一个输入的后缀表达式转换为等价的中缀表达式，

但不带冗余括号。如：输入 $ab*cd++$ ，输出 $a*b+(c+d)$ ；输入 $ab*cd++*e+$ ，输

出 $a*b*(c+d)+e$ 。

属性的含义：

code: 生成的中缀表达式。

op: 当前表达式的符号。

	产生式	语法规则
1)	$L \rightarrow S_n$	<code>printf(Sn.code)</code>
2)	$S_n \rightarrow S_1 S_2 +$	<code>if S2.op=="+" then Sn.code=S1.code "+" (" S2.code ")</code>
		<code>else Sn.code=S1.code "+" S2.code</code>
		<code>Sn.op="+"</code>
3)	$S_n \rightarrow S_1 S_2 *$	<code>if S2.op=="*" then S2.code=(" S2.code ")</code>
		<code>if S1.op=="+" && S2.op=="+" then Sn.code=(" S1.code ") * (" S2.code ")</code>
		<code>else if S1.op=="+" then Sn.code=(" S1.code ") * S2.code</code>
		<code>else if S2.op=="+" then Sn.code=S1.code "*" (" S2.code ")</code>
		<code>else Sn.code=S1.code "*" S2.code</code>
		<code>Sn.op="*"</code>
4)	$S_n \rightarrow id$	<code>Sn.code=id</code>
		<code>Sn.op=""</code>

二.论述题 (共 1 题,33.4 分)

1

(100 分) 如果觉得第一大题的两道题目有点牛刀小试的感觉, 可以考虑玩玩下面的这道题 (5.1 节 PPT 中有) 。

注意: 全部完成第一大题的两道小题即视为完成本次作业, 只完成第二大题同样视为完成本次作业; 两大题都正确完成者可以申请加分 (微信单独申请, 直接加平时成绩的总分) 。大题题号后的分数是系统加的, 不要管它。

❖ (P195) 设计一个SDD, 将一个带有+和*的中缀表达式翻译成没有冗余括号的表达式。比如, 因为两个运算符都是左结合的, 并且*的优先级高于+, 所以

$((a*(b+c))*d)$

可翻译为:

$a*(b+c)*d$

注意: 为了降低难度, 可以无二义的左递归文法作为基础文法进行分析。

参考: <https://github.com/fool2fish/dragon-book-exercise->

使用语法：

```

L -> En
E -> E1 + T
E -> T
T -> T1 * F
T -> F
F -> (E)
F -> digit
    
```

其中 En 包括了 E 和 E1。

属性的含义：

wrapped: 表达式最外层是否有括号。

precedence: 令 +, *, () 和单 digit 的优先级分别为 0, 1, 2, 3。如果表达式最外层有括号，则为去掉括号后最后被计算的运算符的优先级，否则为表达式最后被计算的运算符的优先级。

expr: 表达式。

cleanExpr: 去除了冗余括号的表达式。

	产生式	语法规则
1)	L -> E	L.cleanExpr = E.wrapped ? E.cleanExpr : E.expr; printf(L.cleanExpr)
2)	E -> E1 + T	E.wrapped = false
		E.precedence = 0
		E.expr = E1.expr "+" T.expr
		E.cleanExpr = (E1.wrapped ? E1.cleanExpr : E1.expr) "+" (T.wrapped ? T.cleanExpr : T.expr)
3)	E -> T	E.wrapped = T.wrapped
		E.precedence = T.precedence
		E.expr = T.expr
		E.cleanExpr = T.cleanExpr
4)	T -> T1 * F	T.wrapped = false
		T.precedence = 1
		T.expr = T1.expr "*" F.expr
		T.cleanExpr = (T1.wrapped && T1.precedence >= 1 ? T1.cleanExpr : T1.expr) "*" (F.wrapped && F.precedence >= 1 ? F.cleanExpr : F.expr)
5)	T -> F	T.wrapped = F.wrapped
		T.precedence = F.precedence
		T.expr = F.expr
		T.cleanExpr = F.cleanExpr
6)	F -> (E)	F.wrapped = true
		F.precedence = E.precedence
		F.expr = "(" E.expr ")"
		F.cleanExpr = E.cleanExpr
7)	F -> digit	F.wrapped = false
		F.precedence = 3
		F.expr = digit
		F.cleanExpr = digit