编译原理第五章(三)

李鹏辉

2018 年 11 月 26 日

1.(5.4.4):为下面的产生式写出一个和例5.19类似的L属性SDD。这里的每一个产生式表示一个常见的C语言中那样的控制流结构。你可能需要生成一个三地址语句来跳转到某个标号L，此时你可以生成语句goto L.

1) $S \rightarrow if\ (C)\ S_1\ else\ S_2$

2) $S \rightarrow do\ S_1\ while\ (C)$

3) $S \rightarrow' \{'\ L\ '\}'\ ; L \rightarrow LS \mid \varepsilon$

请注意，列表中的任何简单语句都可能包含一条从它内部跳转到下一条语句的跳转指令，因此简单地每个语句按顺序生成代码是不够的。

1)

$L_1 = new()$

$L_2 = new()$

$L_3 = new()$

$S_1.next = S.next$

$S_2.next = S.next$

$C.false = L_2$

$C.true = L_1$

$S.code = C.code||label||L_1||S_1.code||goto\ \ L_3||label||L_2||S_2.code||lable||L_3$

2)

$L_1 = new()$

$L_2 = new()$

$C.true = L_1$

$C.false = S.next$

$S_1.next = L_2$

$S.code = label||L_1||S1.code||lable||L_2||C.code$

3)

$S.code = L.code$

$L.code = L_1.code||S.code$

2.(5.5.4)按照5.3.3的风格，将5.4.4中得到的每个SDD和一个LL语法分析器一起实现，但是代码风格(或指向代码的指针)存放在栈中

1.

top																	
↓																	
S	**synthesis S.code**																
next = x	code=?																
	data																
top																	
↓																	
if	**(**	**Action**	**C**	**synthesize**	**)**	**S1**	**synthesize**	**else**	**S2**	**synthesize**	**synthesize**						
				C.code		next	S1.code		next	S2.code	S.code						
		snext = x	false = ?	code = ?			code= ?			code = ?							
		L1 = ?	true = ?							Ccode = ?							
		L2 = ?								S1code = ?							
										I1 = ?							
		L1 = new()	stack[top-6].Ccode = code;			stack[top-3].S1code = code;				I2 = ?							
		L2= new()															
										stack[top-1].code = Ccode || label ||L1 || S1code || goto I2|| label || I2 || code							
	stack[top-1].true = L1																
	stack[top-1].false = L2																
	stack[top-4].next = s.next																
	stack[top-7].next = s.next																
	stack[top-8].I1 = L1																
	stack[top-8].I2 = L2																

2.

top									
↓									
S	**synthesis S.code**								
next = x	code=?								
	data								
top									
↓									
do	**action**	**S1**	**sythesis S1**	**while**	**(**	**C**	**synthesis C**	**)**	**synthesis S**
	next	next	code			true	I1		code
	L1					false	I2		
	L2						code		
	L1 = new()	stack[top-4].S1code = code					s1code		
	L2 = new()								
	stack[top-1].next = L2				stack[top-2].code = label || I1 || s1code || label || I2 || ccode				
	stack[top-6].I1 = I1								
	stack[top-6].I2 = I2								
	stack[top-5].true = I1								
	stack[top-5].false = s.next								

3. 消除左递归后得到

$S \rightarrow' \{'L'\}'$

$L \rightarrow L'$

$L' \rightarrow SL'|\varepsilon$

top							
↓							
L	**synthesis L.code**						
next = x	code=?						
	data						
top							
↓							
action	**S**	**synthesis S**	**action**	**L'**	**synthesis L**	**synthesis L'**	
snext = x	next	code	next	next	code	code	
L					scode		
					I		
L = new()	stack[top-3].scode = code						
stack[top-1]next = L							
stack[top-3].next = snext	stack[top-1].next = next						
stack[top-5].I = I	stack[top-1].code = scode ||label|| I||code						
top							
↓							
L	**synthesis L**						
next = x	code						
action	**synthesis L**						
next = x	code = ?						
stack[top-1].code = ""							

3.(5.5.5)按照5.5.4的风格，将5.4.4中得到的每个SDD和一个LR语法分析器一起实现

5.5.4和5.5.5写出翻译方案即可，注意5.5.4的动作位置，而5.5.5参照5.5.4节，把规约动作放在最右端

?	if	(	M1	C	)	M2	S1	else	M3	S2
S.next			C.true	C.code		S1.next	S1.code		S2.next	S2.code
			C.false							
			L1							
			L2							
			L1 =new()		S1.next =stack[top-6].next			S2.next = stack[top-9].next		
			L2= new()							
			L3=new()							
			C.true = L1							
			C.false = L2							
tempcode = stack[top-6].code || label ||stack[top-7].l1 || stack[top-3].code || goto stack[top-10].next = label || stack[top-7].l2 || stack[top].code										
top = top -9										
stack[top].code =tempcode										

?	do	M1	S1	while	(	M2	C	)
S.next = ?		S1.next	S1.code			C.true	C.code	
		L1				C.false		
		L1= new()			C.true=stack[top-4].L1			
		S1.next = L1			C.false=stack[top-6].next			
tempcode = label||stack[top-6].l1||stack[top-5].code||stack[top-1].code								
top = top-7								
stack[top].code = tempcode								

?	{	M1	L	}						
S.next		L.next	L.code							
					tempcode = stack[top-1].code					
	L.next = stack[top-2].next				top = top-3					
					stack[top].cde = tempcode;					
?	M2	L	M3	S						
L.next	L.next	L.code	S.next	S.code						
	L=nw()	S.next = stack[top-3].next								
	L.next = L					tempcode =stack[top-2].code||stack[top].code				
						top=top-3				
						stack[top].code = tempcode				
?										
L.next		tempcode = ""								
		top = top+1								
		stack[top].code = tempcode								