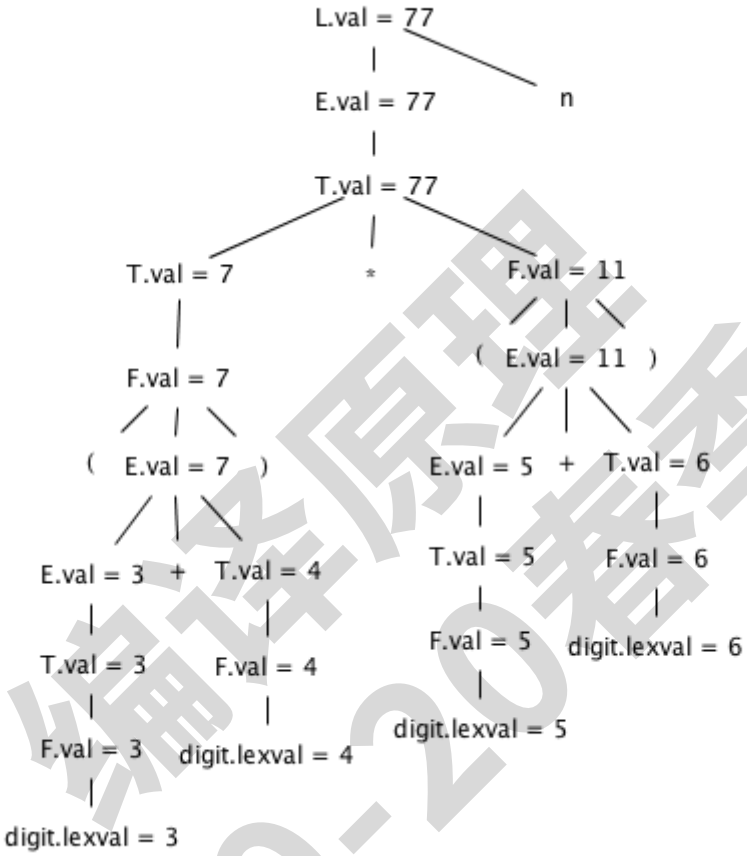
第五章作业

练习 5.1.1



练习 5.1.2

产生式	语义规则
L -> En	L.val = E.val
E -> TE’	E’.inh = T.val E.val = E’.syn
E’ -> +TE_1’	E_1’.inh = E’.inh + T.val E’.syn = E_1’.syn
E’ -> ε	E’.syn = E’.inh
T -> FT’	T’.inh = F.val T.val= T’.syn
T’ -> *FT_1’	T_1’.inh = T’.inh*F.val T’.syn = T_1’.syn
T’ -> ε	T’.syn = T’.inh
F -> (E)	F.val = E.val
F -> digit	F.val = digit.lexval

练习 5.2.3

（1）不是；是；存在
（2）不是；是；存在
（3）是；是；存在
（4）不是；不是；不存在

练习 5.3.1

1)

产生式	语义规则

E→E_1+T	if (E_1.type==integer && T.type==integer){
	E.type = integer
	}else{
	E.type = real
	}

E→T	E.type = T.type
T→**num.num**	T.type = real
T→**num**	T.type = integer

2)

产生式产生式	语义规则
E→E_1+T	if (E_1.type==integer && T.type==integer){ E.type = integer

	E.post = E1.post || T.post || 'int+'
	}else{
	E.type = real
	if (E_1.type==integer){
	E_1.type = real
	E_1.post = E_1.post || "inttoreal"
	}
	if(T.type==integer){
	T.type =real
	T.post = T.post || "inttoreal"
	}
	E.post = E_1.post || T.post || 'float+"
	}

E→T	E.type = T.type E.post = T.post
T→**num.num**	T.type = real T.post = **num.num**
T→**num**	T.type = integer T.post:= **num**

其中 post 属性为后缀符号串，'||'符号为连接运算

练习 5.4.2

A -> 0A’
A' -> {a}BA' | B{b}A' | ε
B -> 1B'
B' -> {c}AB' | A{d}B' | ε

如果 a、b、c、d 涉及到属性计算的话，变换的结果要更复杂一些。

练习 5.4.6

SDD 与 SDT 中修改的部分使用粗体表示。
SDD：

S -> B　　B.ps = 10

B -> B_1B_2　　B_1.ps = B.ps
B_2.ps = B.ps
B.le = B_1.le + B_2.le
B.ht = max(B_1.ht, B_2.ht)
B.dp = max(B_1.dp, B_2.dp)

B -> B_1 sub B_2

B_1.ps = B.ps
B_2.ps = 0.7 * B.ps
B.le = B_1.le + B_2.le
B.ht = max(B_1.ht, B_2.ht - 0.25 * B.ps)
B.dp = max(B_1.dp, B_2.dp + 0.25 * B.ps)

B -> (B_1)

B_1.ps = B.ps
B.le = B_1.le
B.ht = B_1.ht
B.dp = B_1.dp

B -> text

B.le = getLe(B.ps, text.lexval)
B.ht = getHt(B.ps, text.lexval)
B.dp = getDp(B.ps, text.lexval)

SDT：

S ->
B

{ B.ps = 10;}

B ->
B_1
B_2

{B_1.ps = B.ps;}
{B_2.ps = B.ps;}
{**B.le = B_1.le + B_2.le;**
B.ht = max(B_1.ht, B_2.ht);
B.dp = max(B_1.dp, B_2.dp);}

B ->
B_1 sub
B_2

{B_1.ps = B.ps;}
{B_2.ps = 0.7 * B.ps;}
{**B.le = B_1.le + B_2.le;**
B.ht = max(B_1.ht, B_2.ht - 0.25 * B.ps);
B.dp = max(B_1.dp, B_2.dp + 0.25 * B.ps);}

B -> (
B_1)

{B_1.ps = B.ps;}
{**B.le = B_1.le;**
B.ht = B_1.ht
B.dp = B_1.dp;}

B -> text

{**B.le = getLe(B.ps, text.lexval);**
B.ht = getHt(B.ps, text.lexval);
B.dp = getDp(B.ps, text.lexval);}

5.4.4

（1）

S -> if (C) S_1 else S_2 L_1 = newlabel()
L_2 = newlabel()
C.true = L_1
C.false = L_2
S_1.next = S.next
S_2.next = S.next
S.code = C.code || label || L_1 || S_1.code || goto S.next || label || L_2 || S_2.code

（2）

S -> do S_1 while(C) L_1 = newlabel ()
L_2 = newlabel ()
C.true = L_1
C.false = S.next
S_1.next = L_2
S.code = label || L_1 || S_1.code || label || L_2 || C.code

（3）

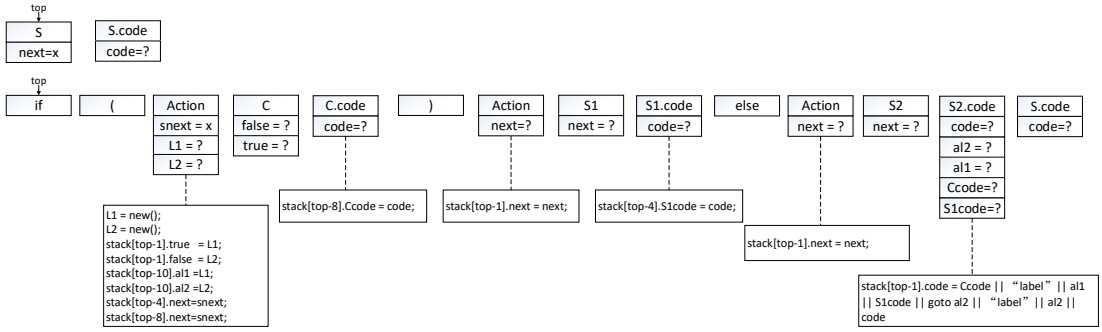
S -> ‘{‘ L ‘}’ L.next = S.next
S.code = L.code

L -> L_1S M = newlabel ()
L_1.next = M
S.next = L.next
L.code = L_1.code || label || M || S_1.code

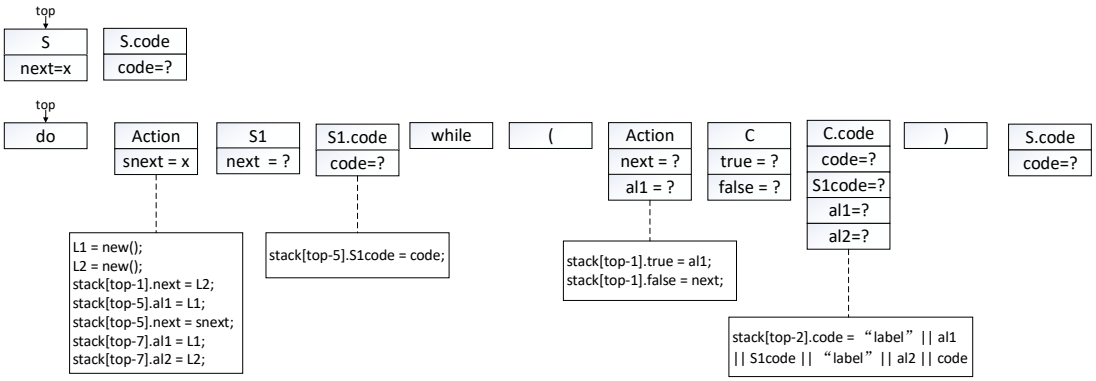
L -> ε L.code = “”

5.5.4：图中用于申请标号的 new()，应该是 newlabel()，请注意。

a)



b)



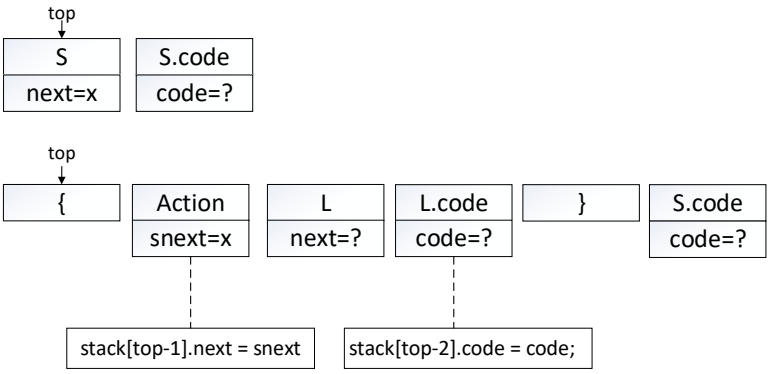
c) 原文法包含左递归，需要先消除左递归。

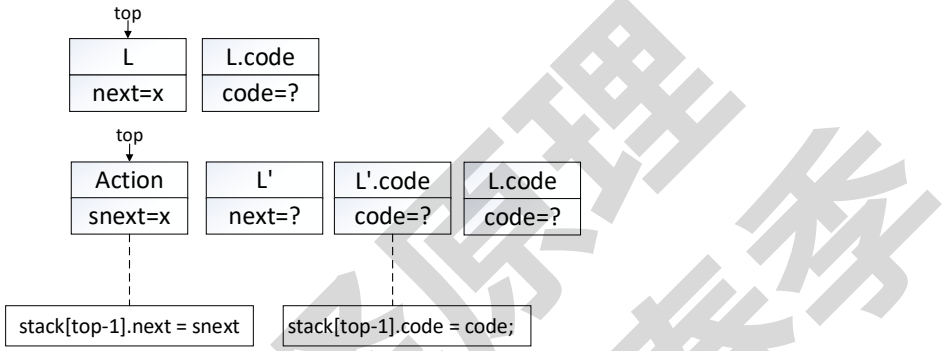
对 L –> LS | ε 消除左递归，得到文法：

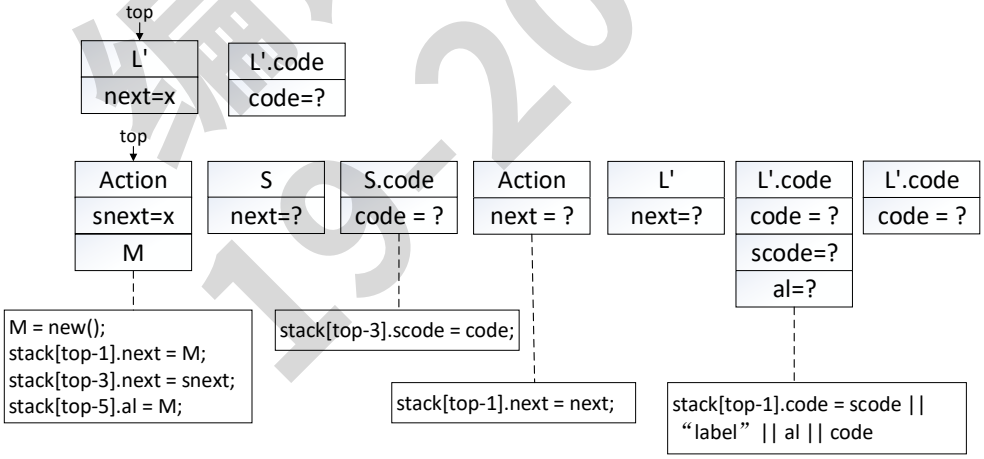
S -> ‘{’ L ‘}’

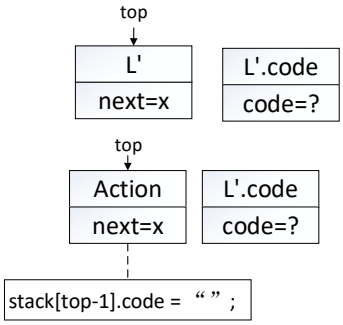
L ->L’

L’ –> SL’ | ε


top
S
next=x
S.code
code=?
top
{
Action
snext=x
L
next=?
L.code
code=?
}
S.code
code=?
stack[top-1].next = snext
stack[top-2].code = code;

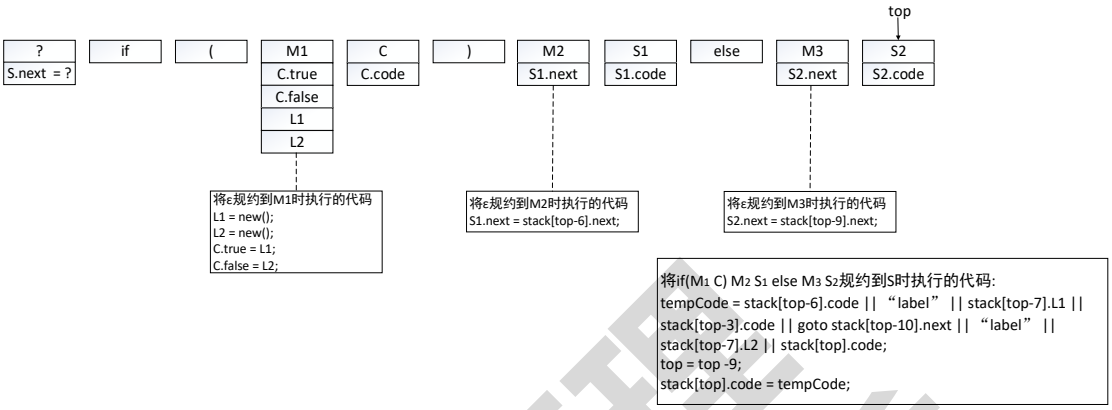

top
L
next=x
L.code
code=?
top
Action
snext=x
L'
next=?
L'.code
code=?
L.code
code=?
stack[top-1].next = snext
stack[top-1].code = code;


top
L'
next=x
L'.code
code=?
top
Action
snext=x
M
S
next=?
S.code
code = ?
Action
next = ?
L'
next=?
L'.code
code = ?
scode=?
al=?
L'.code
code = ?
M = new();
stack[top-1].next = M;
stack[top-3].next = snext;
stack[top-5].al = M;
stack[top-3].scode = code;
stack[top-1].next = next;
stack[top-1].code = scode ||
“label” || al || code

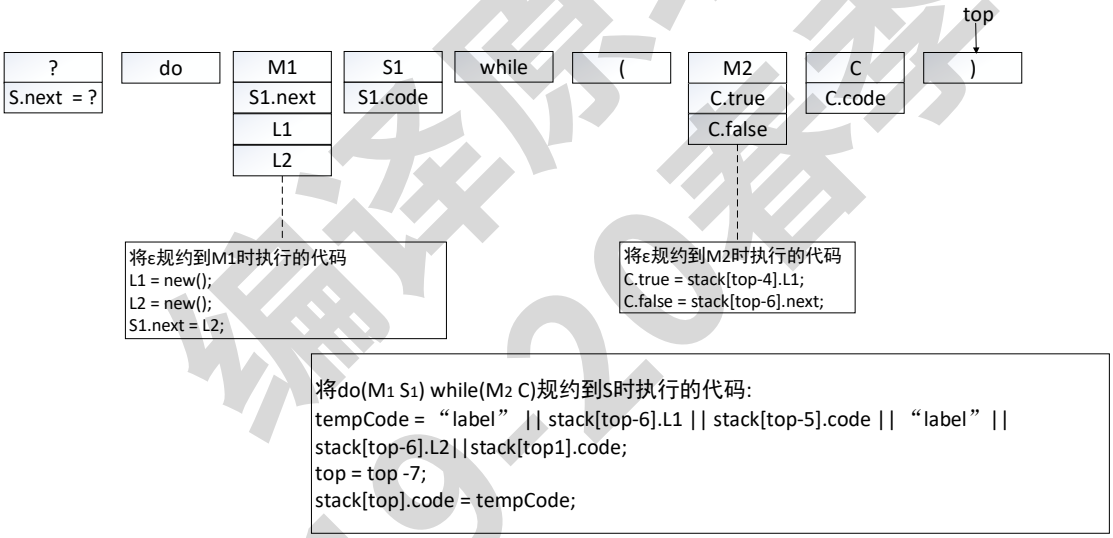

top
L'
next=x
L'.code
code=?
top
Action
next=x
L'.code
code=?
stack[top-1].code = “ ”;

练习 5.5.5 图中用于申请标号的 new()，应该是 newlabel()，请注意。

a)



b)

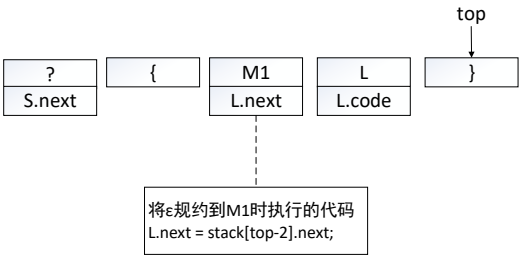


c)原文法包左递归，消除左递归后可得文法：

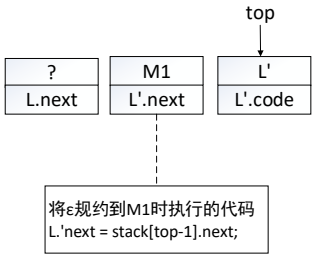
S -> ‘{’ L ’}’

L ->L’

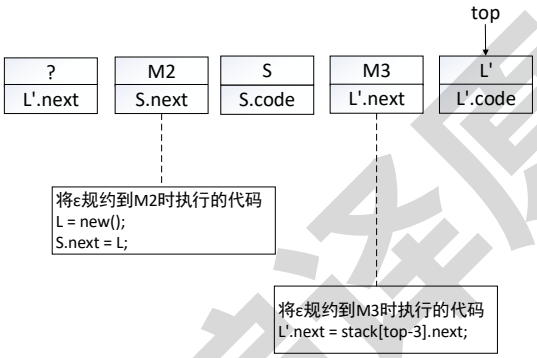
L’ –> SL’ | ε



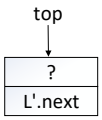
将{M1L}规约到S时执行的代码
tempCode = stack[top-1].code;
top = top － 3;
stack[top].code = tempCode;



将M1L'规约到L时执行的代码
tempCode = stack[top].code;
top = top － 1;
stack[top].code = tempCode;



将M2 S M3 L'规约到L'时执行的代码
tempCode = stack[top-2].code ||"label"||stack[top-3].next||
stack[top].code;
top = top － 3;
stack[top].code = tempCode;



将ε规约到L'时执行的代码
tempCode = “ ”;
top = top + 1;
stack[top].code = tempCode;