编译原理

第七章 语义分析和中间代码产生

第七章 语义分析和中间代码产生

- 中间语言
- 赋值语句的翻译
- 布尔表达式的翻译
- 控制语句的翻译
- 过程调用的处理

第七章 语义分析和中间代码产生

- 中间语言
- 赋值语句的翻译
- 布尔表达式的翻译
- 控制语句的翻译
- 过程调用的处理

7.5 控制语句的翻译

- 考虑下列产生式所定义的语句

$S \rightarrow$ if E then S_1

| if E then S_1 else S_2

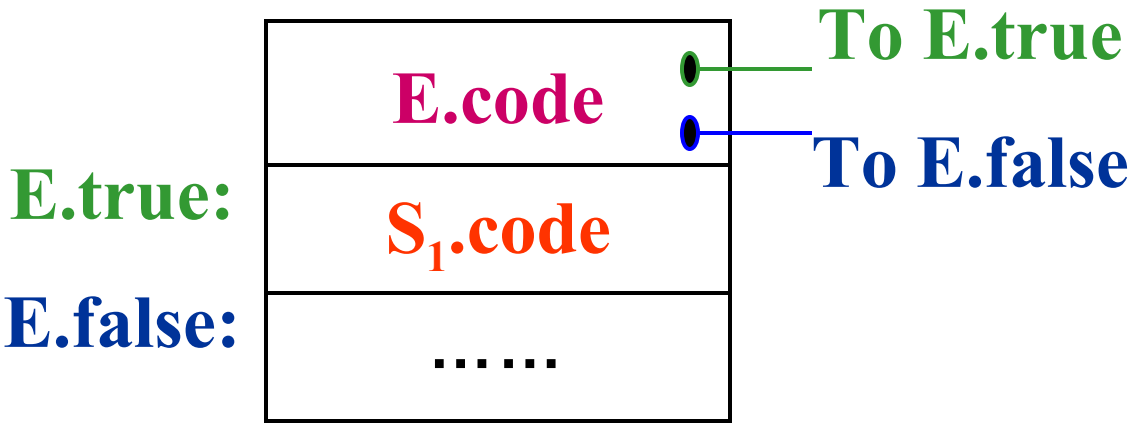
| while E do S_1

其中 E 为布尔表达式。

利用属性文法定义语义
设计一遍扫描的翻译模式

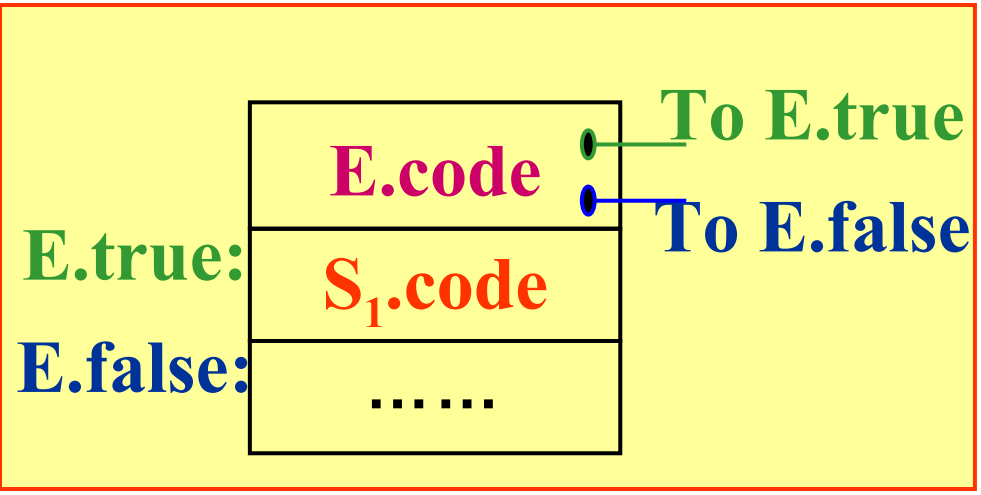
- if-then 语句

  $S \rightarrow$ if E then S_1



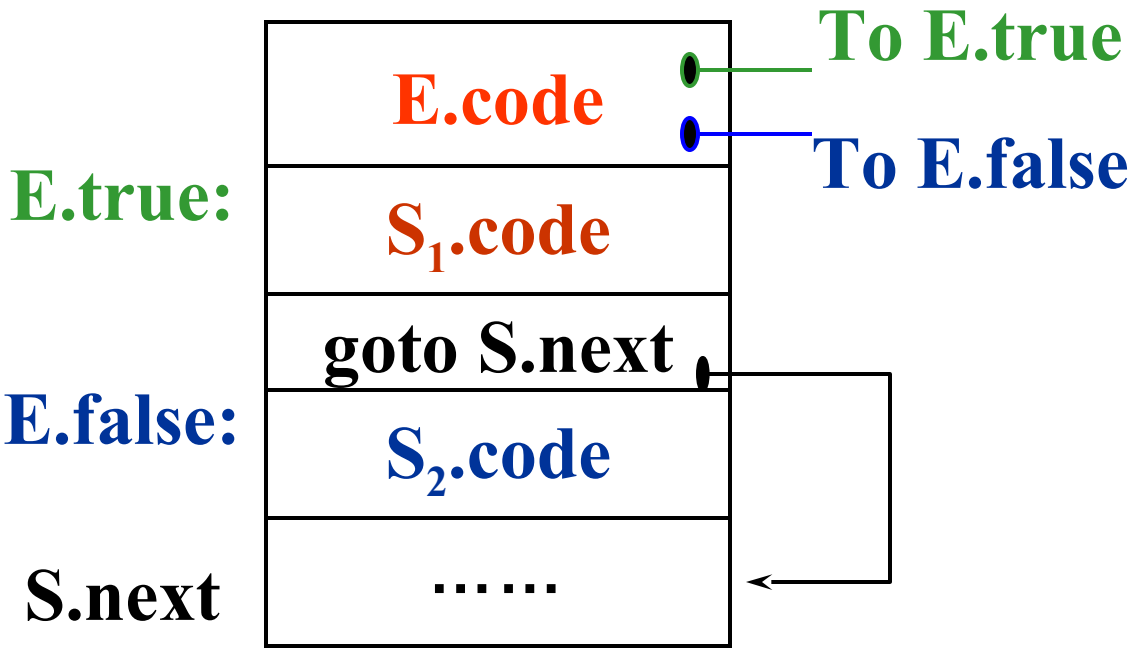
if-then 语句的属性文法

产生式	语义规则
$S \rightarrow$ if E then S_1	E.true:=newlabel; E.flase:=S.next; S_1.next:=S.next S.code:=E.code || gen(E.true ':') || S_1.code



- if-then-else 语句

  $S \rightarrow$ if E then S_1 else S_2



if-then-else 语句的属性文法

产生式

$S \rightarrow$ if E then S_1 else S_2

语义规则

E.true:=newlabel;

E.false:=newlabel;

S_1.next:=S.next

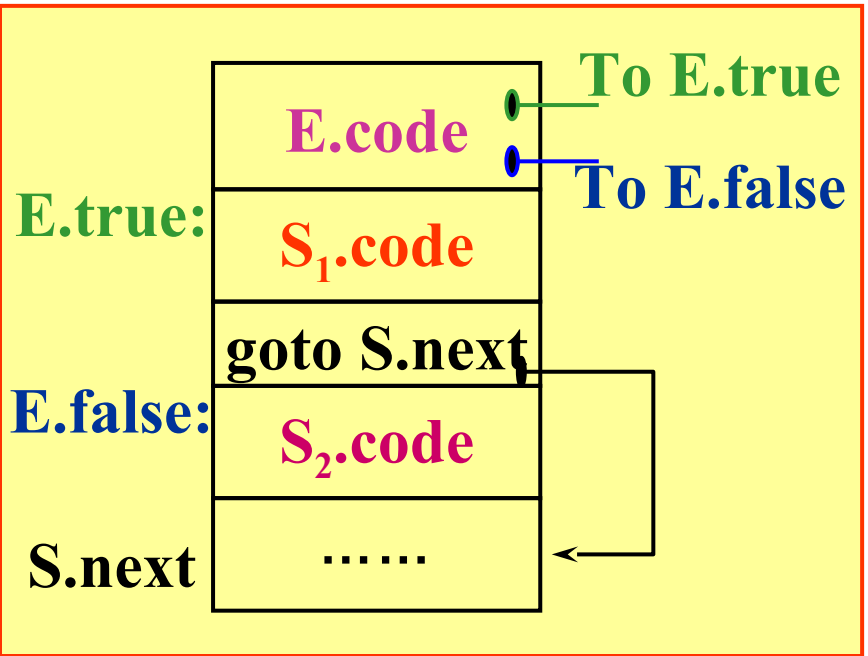
S_2.next:=S.next;

S.code:=E.code ||

gen(E.true ':') || S_1.code ||

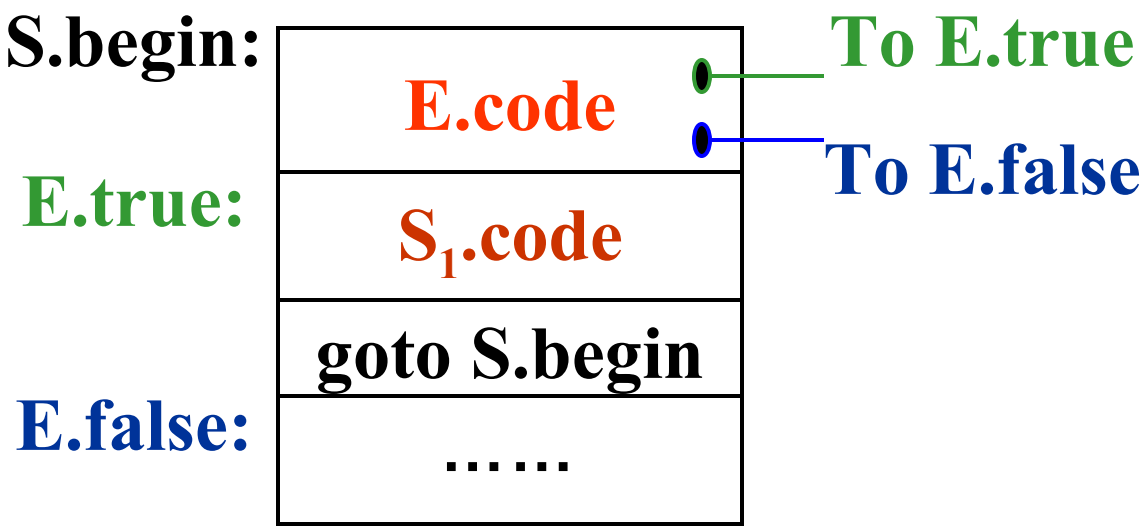
gen('goto' S.next) ||

gen(E.false ':') || S_2.code



- while-do 语句

  $S \rightarrow \text{while } E \text{ do } S_1$



while-do 语句的属性文法

产生式

$S \rightarrow$ while E do S_1

语义规则

S.begin:=newlabel;

E.true:=newlabel;

E.false:=S.next;

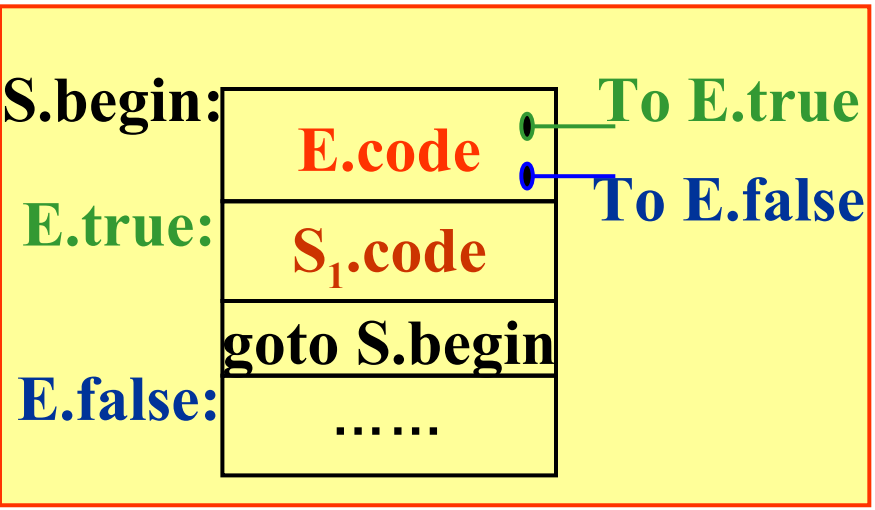
S_1.next:=S.begin;

S.code:=gen(S.begin ':') ||

E.code ||

gen(E.true ':') || S_1.code ||

gen('goto' S.begin)



产生式	语义规则
$S \rightarrow$ if E then S_1	E.true:=newlabel; E.flase:=S.next; S_1.next:=S.next S.code:=E.code || gen(E.true ':') || S_1.code
$S \rightarrow$ if E then S_1 else S_2	E.true:=newlabel; E.false:=newlabel; S_1.next:=S.next S_2.next:=S.next; S.code:=E.code || gen(E.true ':') || S_1.code || gen('goto' S.next) || gen(E.false ':') || S_2.code
$S \rightarrow$ while E do S_1	S.begin:=newlabel; E.true:=newlabel; E.false:=S.next; S_1.next:=S.begin; S.code:=gen(S.begin ':') || E.code || gen(E.true ':') ||

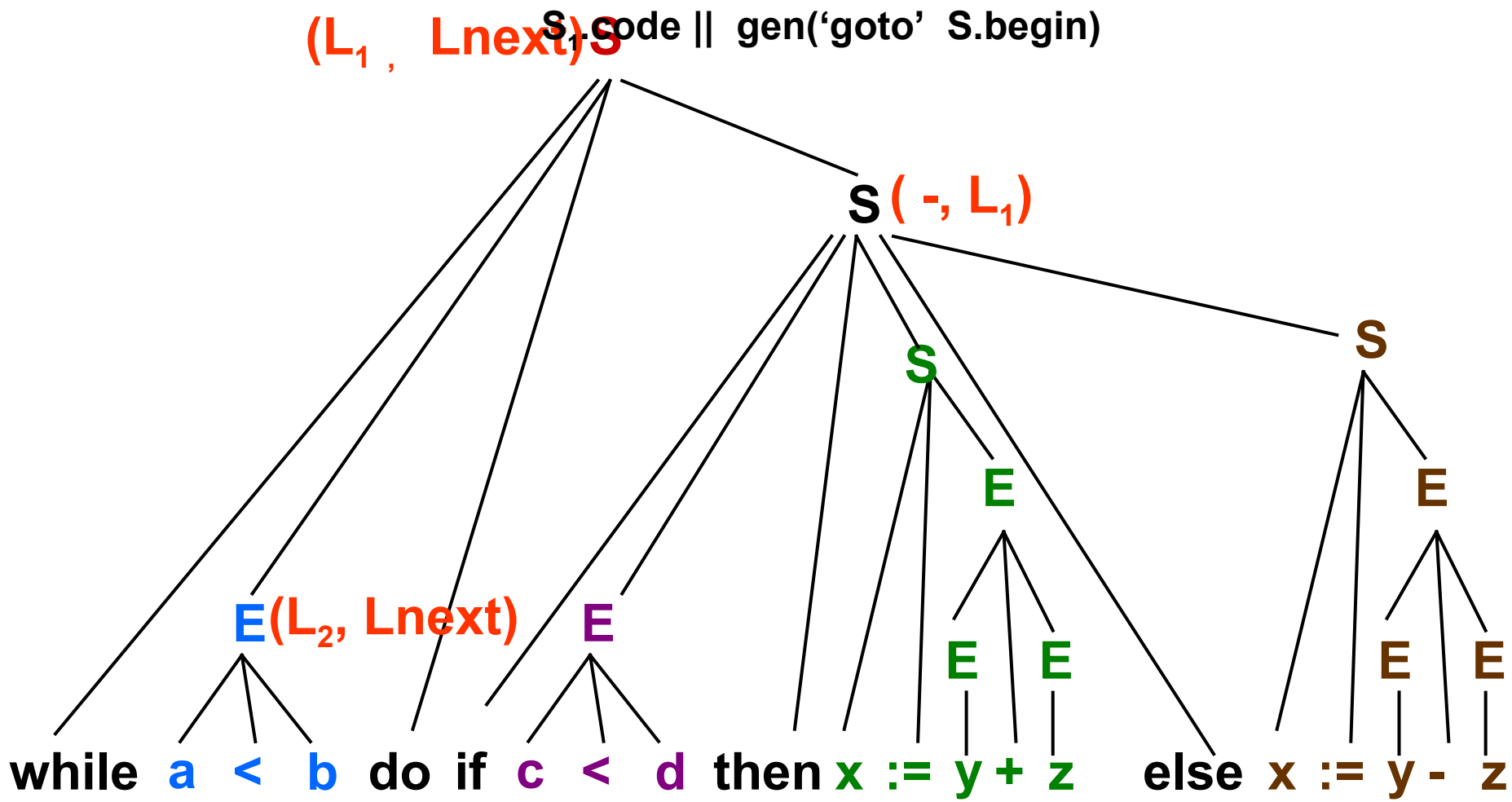
$S \rightarrow$ while E do S_1 S.begin:=newlabel;

E.true:=newlabel;

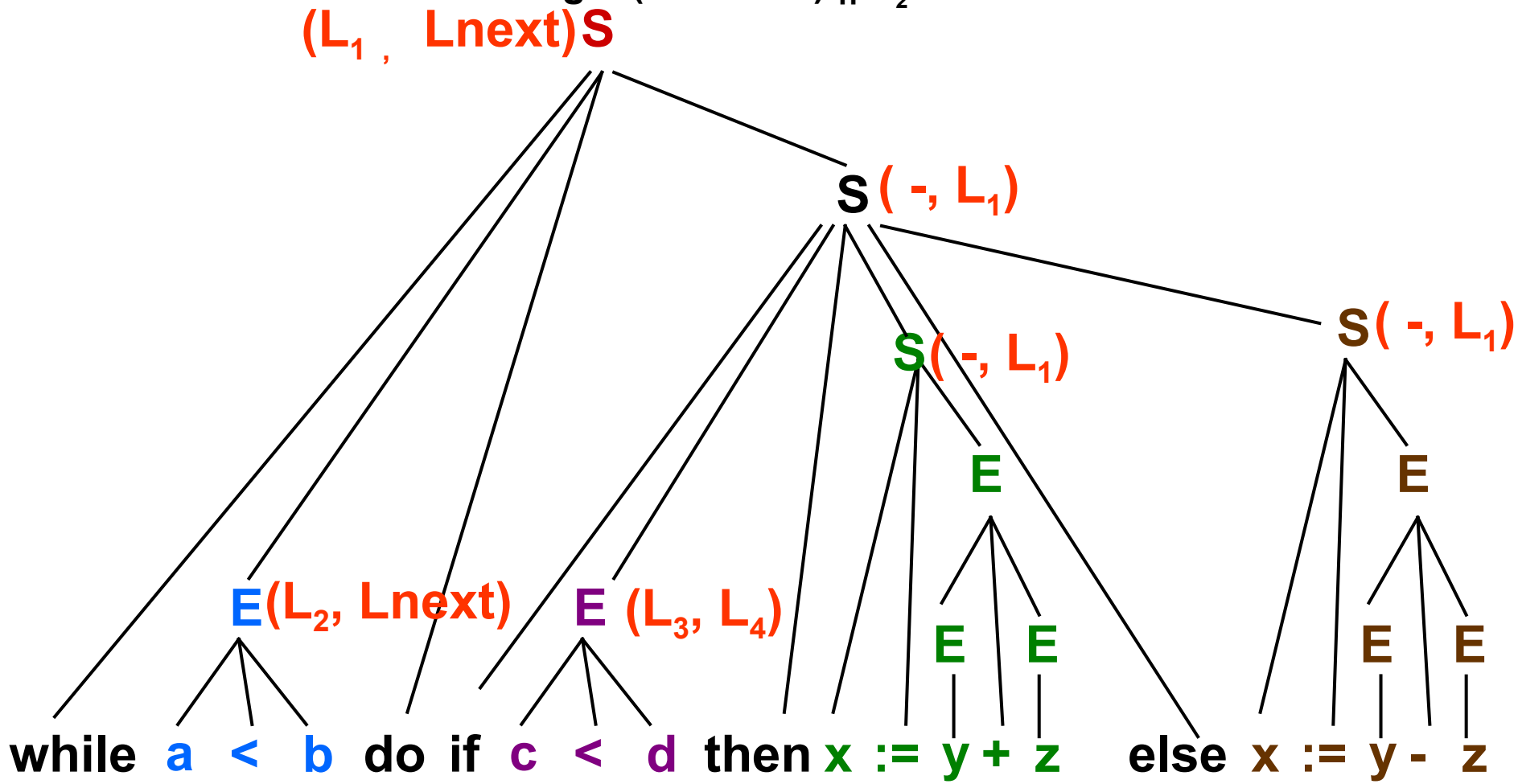
E.false:=S.next;

S_1.next:=S.begin;

S.code:=gen(S.begin ':') || E.code || gen(E.true ':') ||

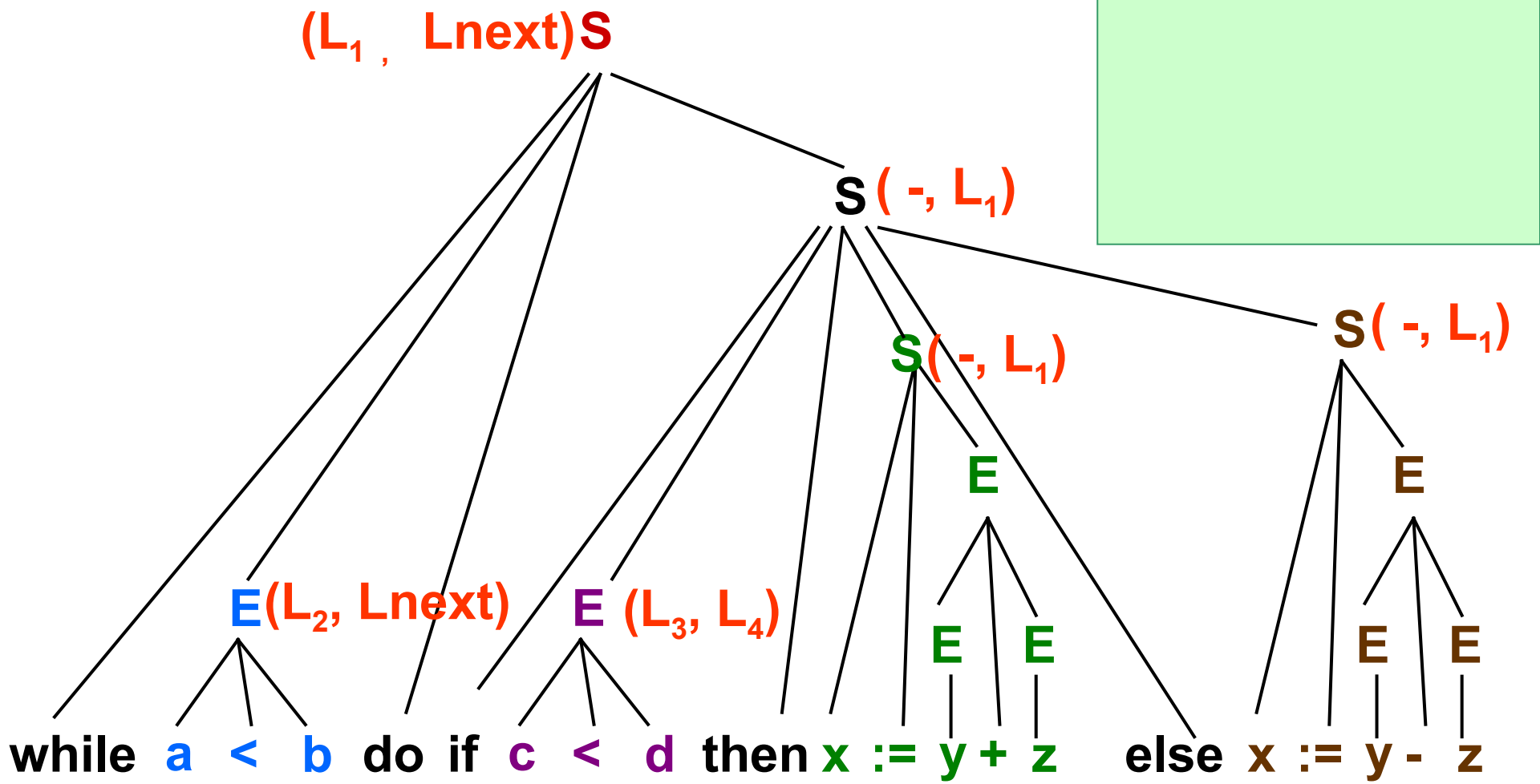


$S \rightarrow$ if E then S_1 else S_2 E.true:=newlabel;
E.false:=newlabel;
S_1.next:=S.next
S_2.next:=S.next;
S.code:=E.code || gen(E.true ':') ||
S_1.code || gen('goto' S.next) ||
gen(E.false ':') || S_2.code



产生式	语义规则
$E \rightarrow id_1\ relop\ id_2$	E.code:=gen('if ' id_1.place relop.op id_2.place 'goto' E.true) || gen('goto' E.false)

if a<b goto L_2
goto Lnext
if c<d goto L_3
goto L_4



$S \rightarrow id := E$ S.code:=E.code || gen(id.place ':=' E.place)

$E \rightarrow E_1 + E_2$ E.place:=newtemp; E.code:=E_1.code || E_2.code || gen(E.place ':=' E_1.place '+' E_2.place)

$(L_1, Lnext)S$

$S(-, L_1)$

$E(L_2, Lnext)$ $E(L_3, L_4)$

$S(-, L_1)$ $S(-, L_1)$

E E E E E E

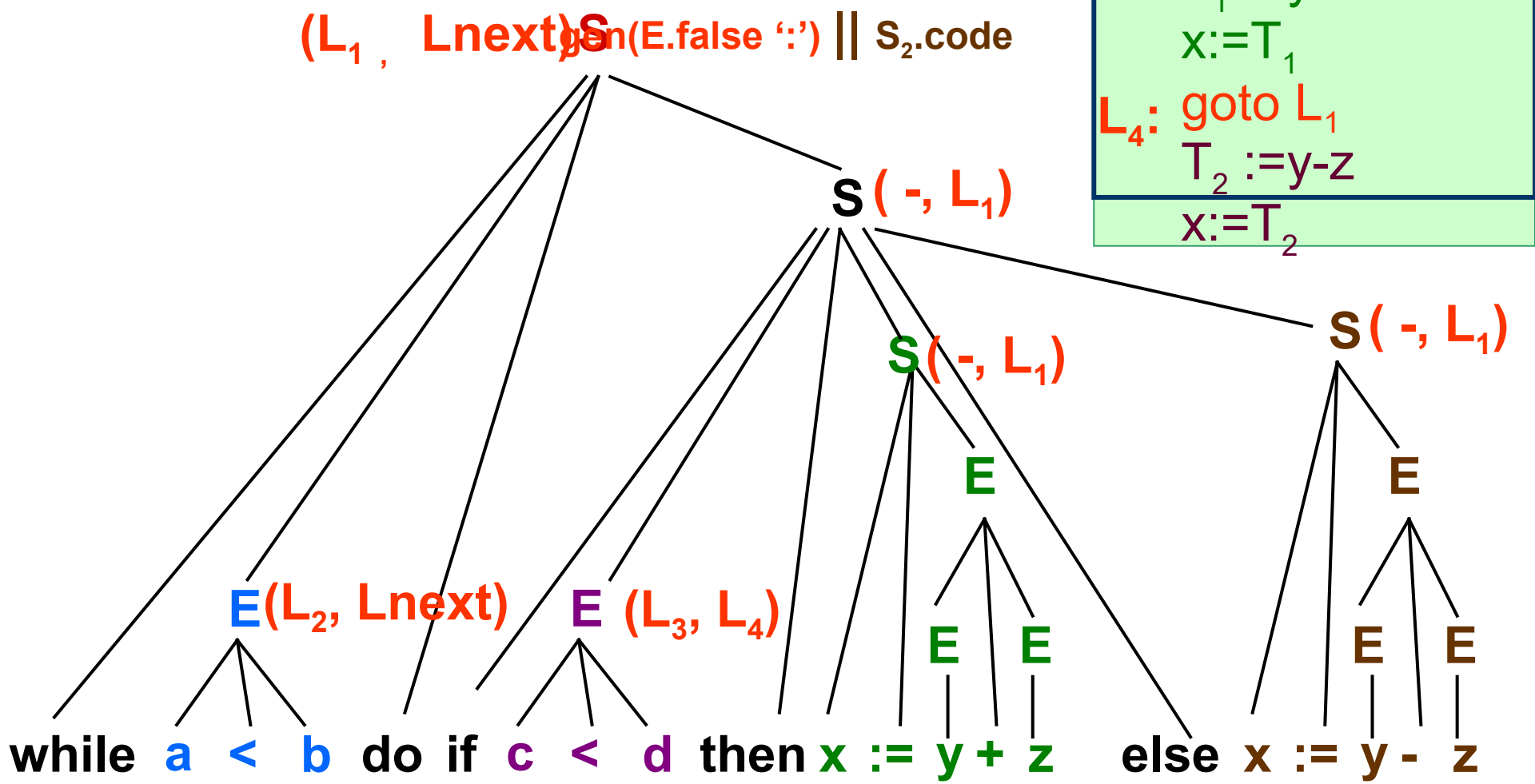
while a < b do if c < d then x := y + z else x := y - z

if a<b goto L_2
goto Lnext
if c<d goto L_3
goto L_4
$T_1 := y+z$
$x := T_1$

$T_2 := y-z$
$x := T_2$

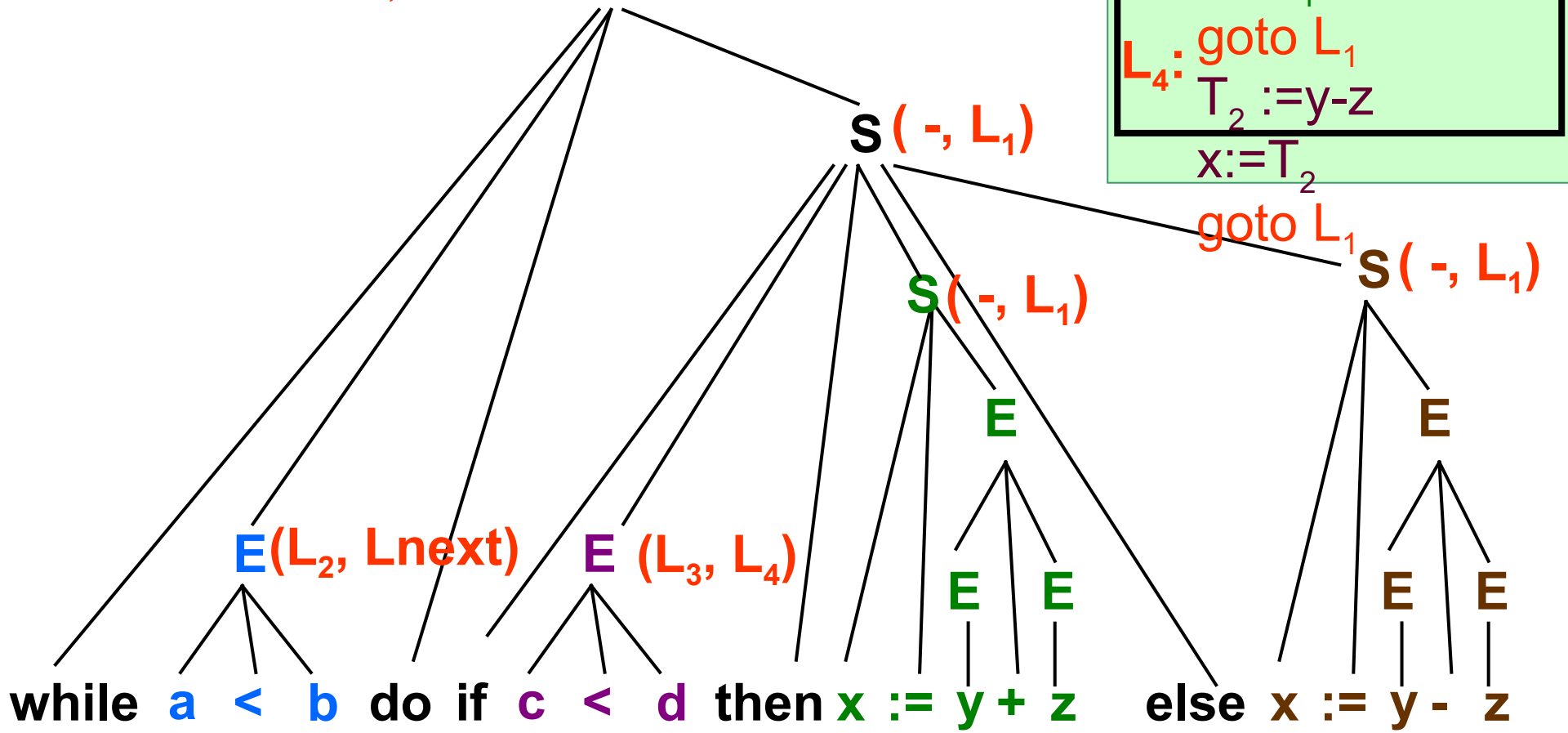
$S \rightarrow$ if E then S_1 else S_2 E.true:=newlabel;
E.false:=newlabel;
S_1.next:=S.next
S_2.next:=S.next;
S.code:=E.code || gen(E.true ':') ||
S_1.code || gen('goto' S.next) ||
S_2.code || gen(E.false ':')

```
     if a<b goto L2
     goto Lnext
     if c<d goto L3
     goto L4
L3:  T1 :=y+z
     x:=T1
L4:  goto L1
     T2 :=y-z
     x:=T2
```



$S \rightarrow$ while E do S_1 S.begin:=newlabel;
E.true:=newlabel;
E.false:=S.next;
S_1.next:=S.begin;
S.code:=gen(S.begin ':') || E.code ||
gen(E.true ':') || S_1.code || gen('goto' S.begin)

L_1: if a<b goto L_2
goto Lnext
L_2: if c<d goto L_3
goto L_4
L_3: T_1 :=y+z
x:=T_1
L_4: goto L_1
T_2 :=y-z
x:=T_2
goto L_1



考虑如下语句：

```
while a<b do
        if c<d then  x:=y+z
                else   x:=y-z
```

- 生成下列代码：

L_1: if a<b goto L_2
goto Lnext
L_2: if c<d goto L_3
goto L_4
L_3: T_1:=y+z
x:=T_1
goto L_1
L_4: T_2:=y-z
x:=T_2
goto L_1

一遍扫描翻译控制流语句

- 考虑下列产生式所定义的语句：

  (1) S→if E then S
  (2) | if E then S else S
  (3) | while E do S
  (4) | begin L end
  (5) | A
  (6) L→L;S
  (7) | S

- S 表示语句，　L 表示语句表，
  A 为赋值语句，E 为一个布尔表达式

- if 语句的翻译

相关产生式

$S \rightarrow$ if E then $S^{(1)}$

| if E then $S^{(1)}$ else $S^{(2)}$

改写后产生式

$S \rightarrow$ if E then M S_1

$S \rightarrow$ if E then M_1 S_1 N else M_2 S_2

$M \rightarrow \varepsilon$

$N \rightarrow \varepsilon$

翻译模式：

1. $S \rightarrow$ if E then M S_1

{ backpatch(E.truelist, M.quad);
S.nextlist:=merge(E.falselist, S_1.nextlist) }

2. $S \rightarrow$ if E then M_1 S_1 N else M_2 S_2

{ backpatch(E.truelist, M_1.quad);
backpatch(E.falselist, M_2.quad);
S.nextlist:=merge(S_1.nextlist, N.nextlist, S_2.nextlist) }

3. $M \rightarrow \varepsilon$ { M.quad:=nextquad }

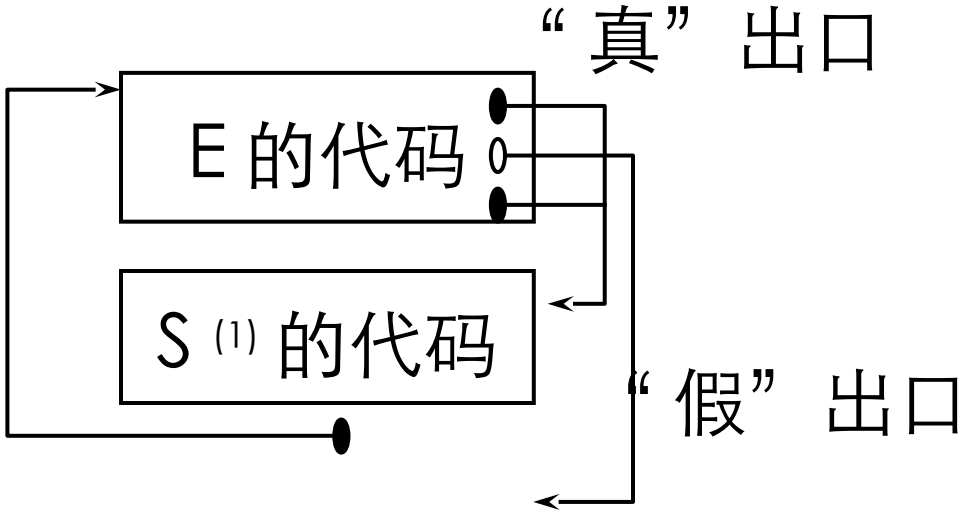
4. $N \rightarrow \varepsilon$ { N.nextlist:=makelist(nextquad);
emit(‘j, – , – , – ’) }

- while 语句的翻译

相关产生式

$S \rightarrow$ while E do $S^{(1)}$

- 翻译为：



为了便于 " 回填 "，改写产生式为：

$S \rightarrow$ while M_1 E do M_2 S_1

$M \rightarrow \varepsilon$

- 翻译模式：

1. $S \rightarrow \text{while } M_1\ E \text{ do } M_2\ S_1$

{

backpatch(E.truelist, M_2.quad);

backpatch(S_1.nextlist, M_1.quad);

S.nextlist:=E.falselist;

emit(‘j, – , – ,’ M_1.quad) }

2. $M \rightarrow \varepsilon$ { M.quad:=nextquad }

- 产生式

$$L \rightarrow L;S$$

改写为:

$$L \rightarrow L_1; M\ S$$

$$M \rightarrow \varepsilon$$

- 翻译模式：

1. $L \rightarrow L_1$; M S
   { backpatch(L_1.nextlist, M.quad);
   L.nextlist:=S.nextlist }

2. $M \rightarrow \varepsilon$
   { M.quad:=nextquad }

- 其它几个语句的翻译

1. S→begin L end
   { S.nextlist:=L.nextlist }

2. S→A
   { S.nextlist:=makelist( ) }

3. L→S
   { L.nextlist:=S.nextlist }

示例：翻译语句

- 将下面的语句翻译为四元式

```
while (a<b) do
        if (c<d) then x:=y+z;
```

P195

S→if E then M S_1

{ backpatch(E.truelist, M.quad)
S.nextlist:=merge(E.falselist, S_1.nextlist) }

M→ε { M.quad:=nextquad }

S→A { S.nextlist:=makelist( ) }

P195

S→while M_1 E do M_2 S_1

{ backpatch(E.truelist, M_2.quad);
backpatch(S_1.nextlist, M_1.quad);
S.nextlist:=E.falselist
emit('j, − , − ,' M_1.quad) }

M→ε { M.quad:=nextquad }

P180

(5) E→id_1 relop id_2

{ E.truelist:=makelist(nextquad);
E.falselist:=makelist(nextquad+1);
emit('j' relop.op ',' id $_1$.place ',' id $_2$.place',' '0');
emit('j, − , − , 0') }

P179

S→id:=E { p:=lookup(id.name);
if p≠nil then emit(p ':=' E.place)
else error }

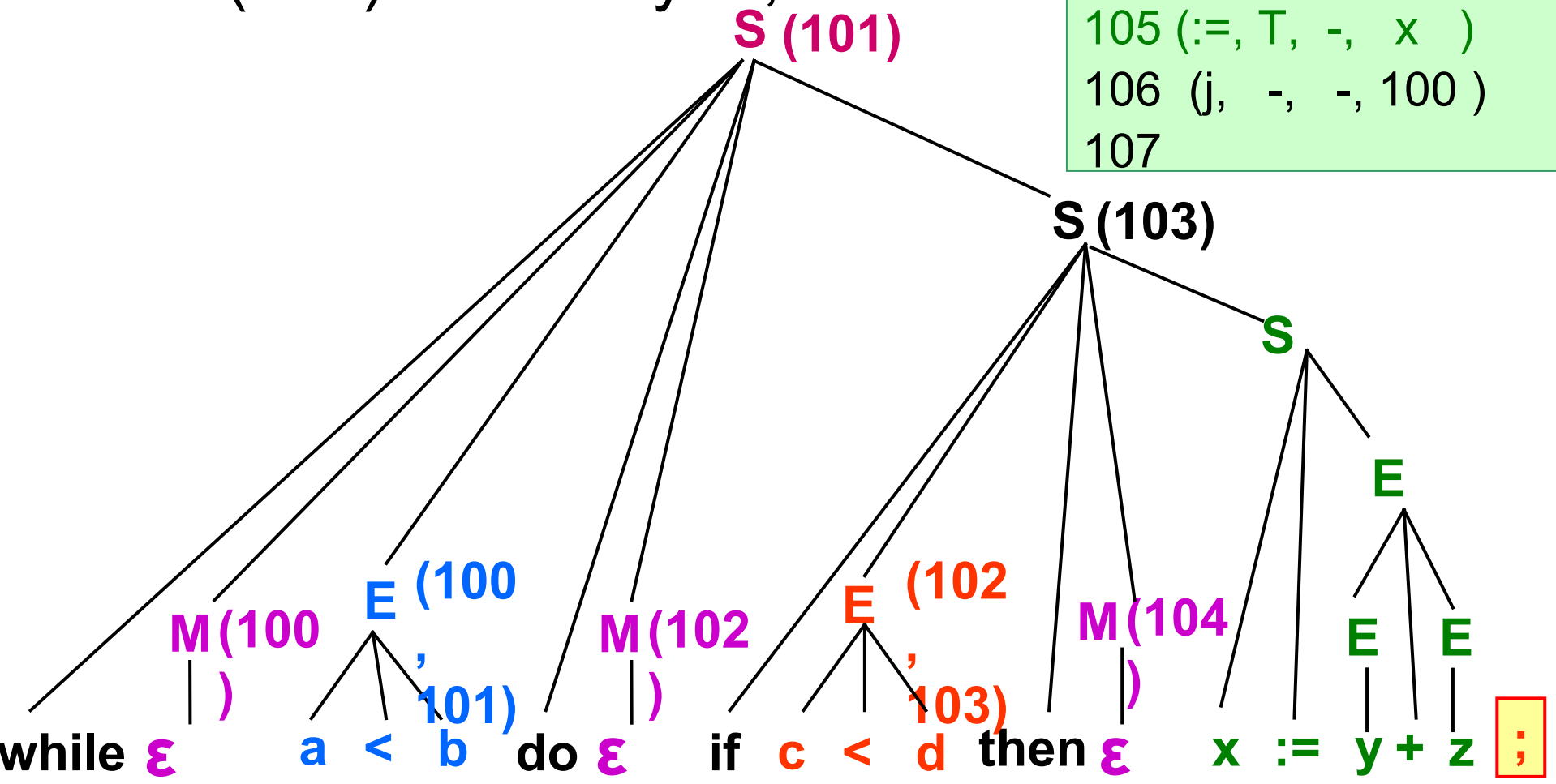
E→E_1+E_2 { E.place:=newtemp;
emit(E.place ':=' E_1.place '+' E_2.place)}

翻译语句

while (a<b) do

if (c<d) then x:=y+z;

100 (j<, a, b, 102)
101 (j, -, -, 107)
102 (j<, c, d, 104)
103 (j, -, -, 100)
104 (+, y, z, T)
105 (:=, T, -, x)
106 (j, -, -, 100)
107



翻译语句

```
while (a<b) do
        if (c<d) then x:=y+z;
```

```
100   (j<, a, b, 102)
101   (j, -, -, 107)
102   (j<, c, d, 104)
103   (j, -, -, 100)
104   (+, y, z, T)
105   (:=, T, -, x)
106   (j, -, -, 100)
107
```

小结

- 控制语句的翻译

$$S \rightarrow \text{if } E \text{ then } S_1$$

$$\mid \text{if } E \text{ then } S_1 \text{ else } S_2$$

$$\mid \text{while } E \text{ do } S_1$$

  - 一遍扫描的翻译模式

作业

- P218 - 7， 12