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

布尔表达式的翻译

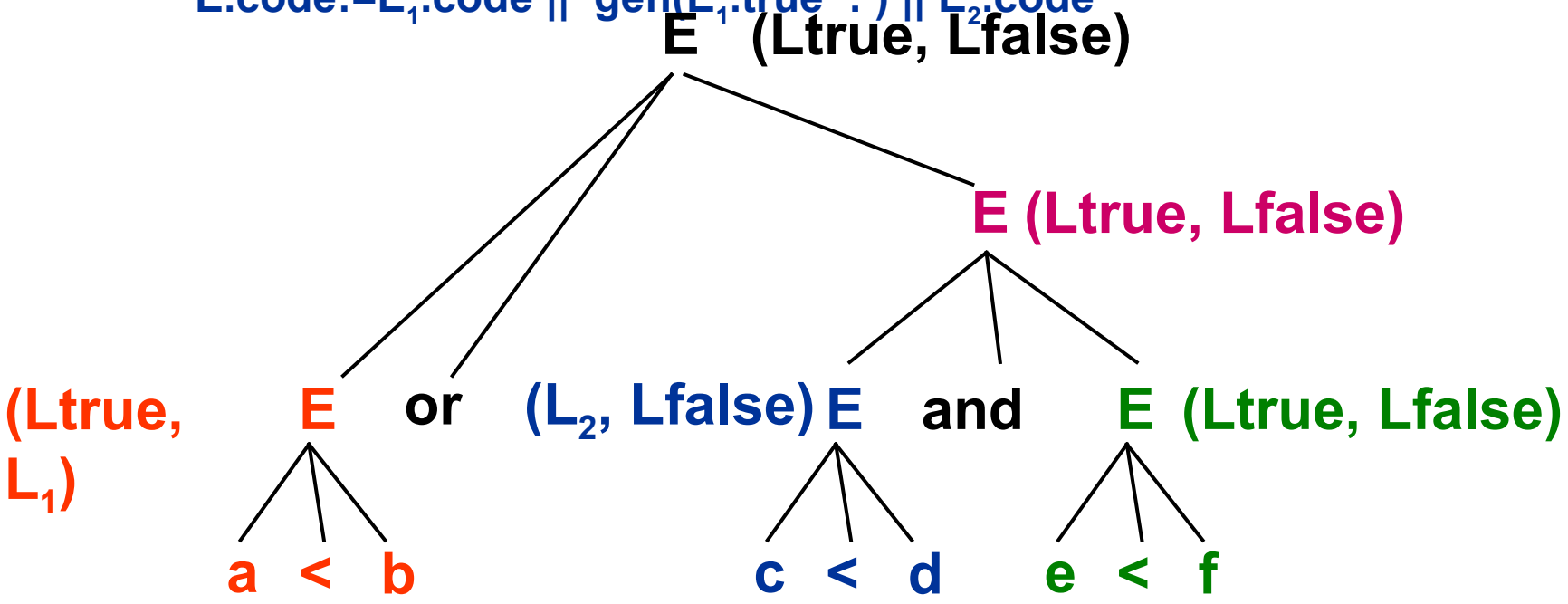
- 两 (多) 遍扫描
  - 为给定的输入串构造一棵语法树
  - 遍历语法树，进行语义规则中规定的翻译

考虑如下表达式：

a<b or c<d and e<f

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)
$E \rightarrow E_1$ or E_2	E_1.true:=E.true; E_1.false:=newlabel; E_2.true:=E.true; E_2.false:=E.false; E.code:=E_1.code || gen(E_1.false ‘:’) || E_2.code
$E \rightarrow E_1$ and E_2	E_1.true:=newlabel; E_1.false:=E.false; E_2.true:=E.true; E_2.false:=E.fasle;

产生式	语义规则
$E \rightarrow E_1$ or E_2	E_1.true:=E.true; E_1.false:=newlabel; E_2.true:=E.true; E_2.false:=E.false; E.code:=E_1.code || gen(E_1.false ':') || E_2.code
$E \rightarrow E_1$ and E_2	E_1.true:=newlabel; E_1.false:=E.false; E_2.true:=E.true; E_2.false:=E.fasle; E.code:=E_1.code || gen(E_1.true ':') || E_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)*
$E \rightarrow E_1$ or E_2	E_1.true:=E.true; E_1.false:=newlabel; E_2.true:=E.true; E_2.false:=E.false; *$E.code:=E_1.code \|\| gen(E_1.false ':') \|\| E_2.code$*
$E \rightarrow E_1$ and E_2	E_1.true:=newlabel; E_1.false:=E.false; E_2.true:=E.true; E_2.false:=E.fasle; *$E.code:=E_1.code \|\| gen(E_1.true ':') \|\| E_2.code$*

if a<b goto Ltrue

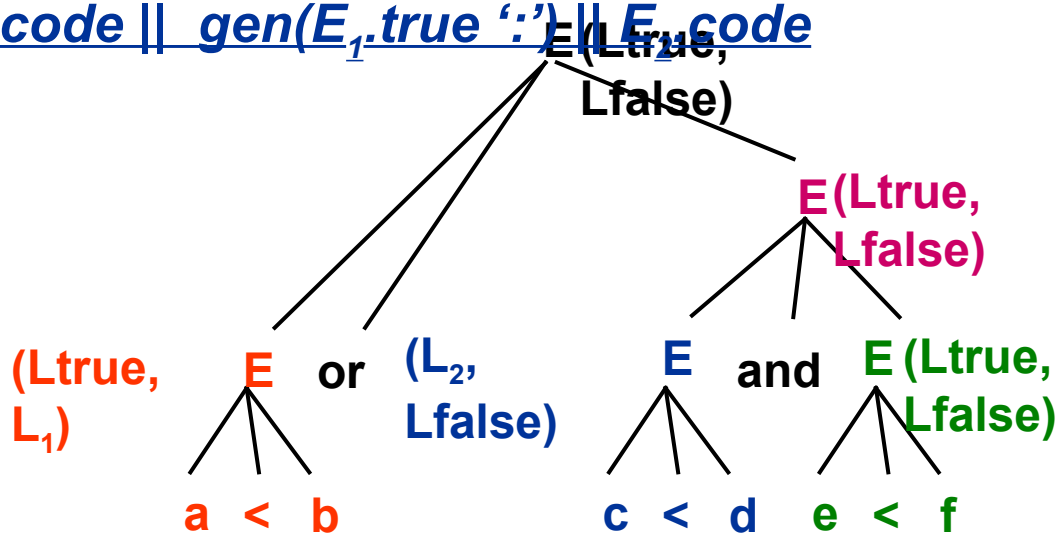
goto L_1

L_1: if c<d goto L_2

goto Lfalse

L_2: if e<f goto Ltrue

goto Lfalse



布尔表达式的翻译

- 两 (多) 遍扫描
  - 为给定的输入串构造一棵语法树
  - 遍历语法树，进行语义规则中规定的翻译
- 一遍扫描

一遍扫描实现布尔表达式的翻译

- 采用四元式形式
- 把四元式存入一个数组中，数组下标就代表四元式的标号
- 约定

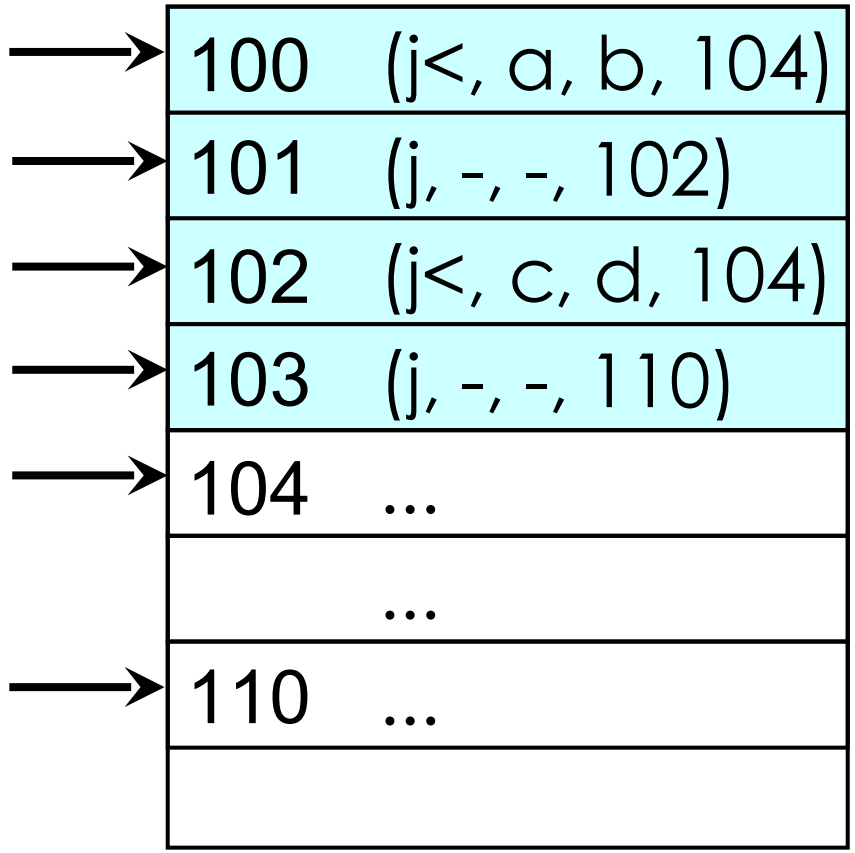
  四元式 (jnz, a, -, p)　表示　if a goto p

  四元式 (jrop, x, y, p)　表示　if x rop y goto p

  四元式 (j, -, -, p)　表示　goto p

一遍扫描实现布尔表达式的翻译

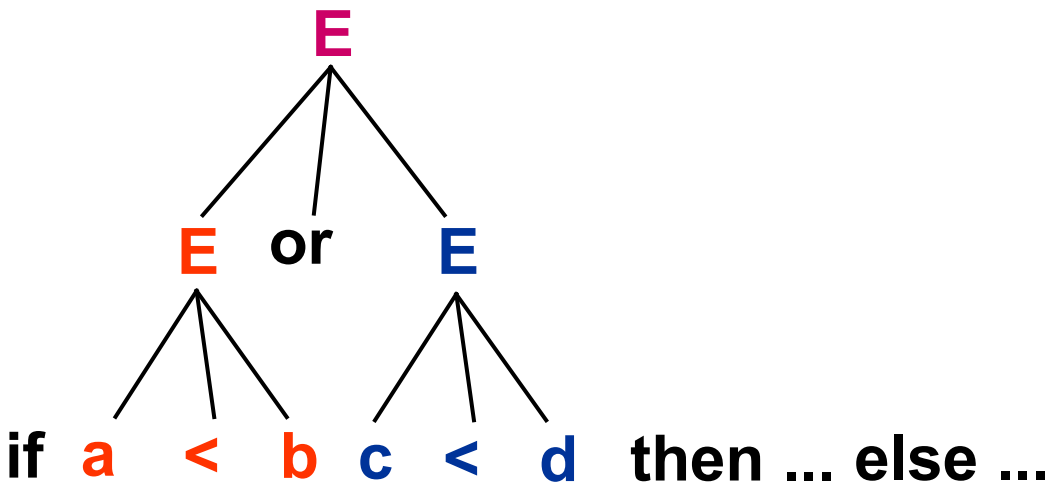
- 过程 emit 将四元式代码送到输出文件中



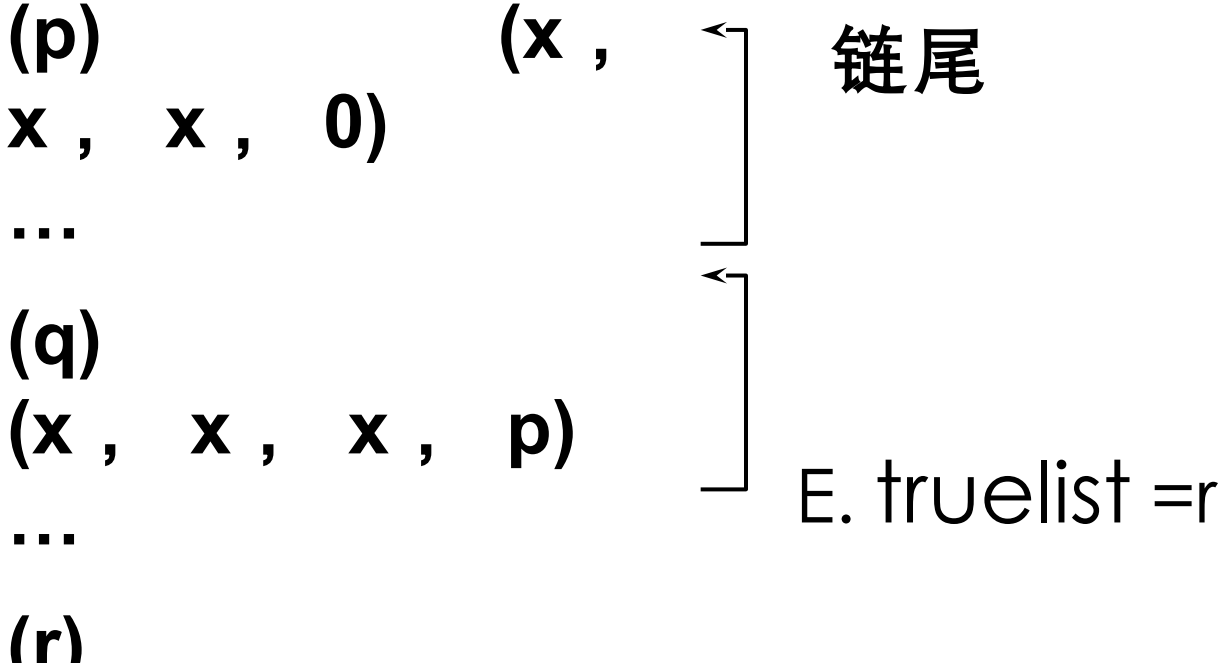
a<b or c<d

- 最大的困难？
  - 产生跳转四元式时，它的转移地址无法立即知道
  - 需要以后扫描到特定位置时才能回过头来确定
- 把这个未完成的四元式地址作为 E 的语义值保存，待机“回填”

100	(j<, a, b, 104)
101	(j, -, -, 102)
102	(j<, c, d, 104)
103	(j, -, -, 110)
104	...
	...
110	...



- 为非终结符 E 赋予两个综合属性 E.truelist 和 E.falselist 。它们分别记录布尔表达式 E 所应的四元式中需回填“真”、“假”出口的四元式的标号所构成的链表
- 例如：假定 E 的四元式中需要回填 " 真 " 出口的 p， q， r 三个四元式，则 E.truelist 为下列链：



- 为了处理 E.truelist 和 E.falselist ，引入下列语义变量和过程
  - 变量 nextquad，它指向下一条将要产生但尚未形成的四元式的地址 (标号)。 nextquad 的初值为 1，每当执行一次 emit 之后， nextquad 将自动增 1。
  - 函数 makelist(i)，它将创建一个仅含 i 的新链表，其中 i 是四元式数组的一个下标 (标号)；函数返回指向这个链的指针。
  - 函数 merge(p_1,p_2)， 把以 p_1 和 p_2 为链首的两条链合并为一，作为函数值，回送合并后的链首。
  - 过程 backpatch(p, t)，其功能是完成“回填”，把 p 所链接的每个四元式的第四区段都填为 t。

过程 backpatch(p, t)，其功能是完成“回填”，把 p 所链接的每个四元式的第四区段都填为 t。

(r)　　(t，
x，　x，　0)
…　　t
(q)
(x，　x，　x，　t)
…
(p)
(x，　x，　x，　q)

布尔表达式的文法

(1) $E \rightarrow E_1$ or M E_2

(2) | E_1 and M E_2

(3) | not E_1

(4) | (E_1)

(5) | id_1 relop id_2

(6) | id

(7) $M \rightarrow \varepsilon$

布尔表达式的翻译模式

(7) $M \rightarrow \varepsilon$ { M.quad:=nextquad }

(1) $E \rightarrow E_1$ or M E_2

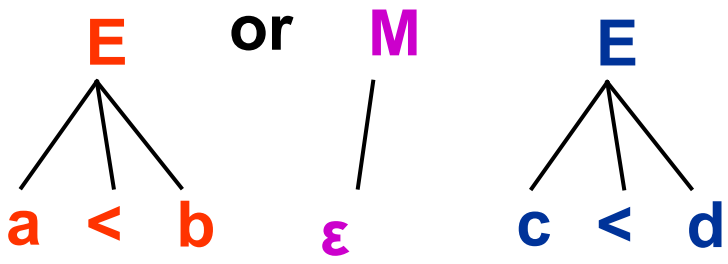
(2) | E_1 and M E_2

(3) | not E_1

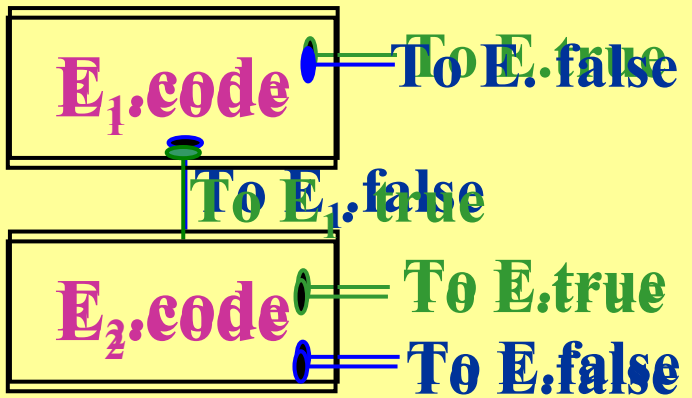
(4) | (E_1)

(5) | id_1 relop id_2

(6) | id



布尔表达式的翻译模式



(1) $E \rightarrow E_1$ or M E_2

{ backpatch(E_1.falselist, M.quad);

E.truelist:=merge(E_1.truelist, E_2.truelist);

E.falselist:=E_2.falselist }

(2) $E \rightarrow E_1$ and M E_2

{ backpatch(E_1.truelist, M.quad);

E.truelist:=E_2.truelist;

E.falselist:=merge(E_1.falselist,E_2.falselist) }

布尔表达式的翻译模式

(3) $E \rightarrow \text{not } E_1$

{ E.truelist:=E_1.falselist;

E.falselist:=E_1.truelist}

(4) $E \rightarrow (E_1)$

{ E.truelist:=E_1.truelist;

E.falselist:=E_1. falselist}

布尔表达式的翻译模式

(5) $E \rightarrow id_1$ 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') }

(6) E→id

{ E.truelist:=makelist(nextquad);
E.falselist:=makelist(nextquad+1);
emit('jnz' ',' id .place ',' ' – ' ' ,' ' 0') ;
emit(' j, -, -, 0') }

布尔表达式的翻译模式

- 作为整个布尔表达式的“真” “假”出口(转移目标)仍待回填.

a<b or c<d and e<f

(5) $E \rightarrow id_1$ 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') }

(7) $M \rightarrow \varepsilon$ { M.quad:=nextquad }

(1) $E \rightarrow E_1$ or M E_2

{ backpatch(E_1.falselist, M.quad);
E.truelist:=merge(E_1.truelist, E_2.truelist);
E.falselist:=E_2.falselist }

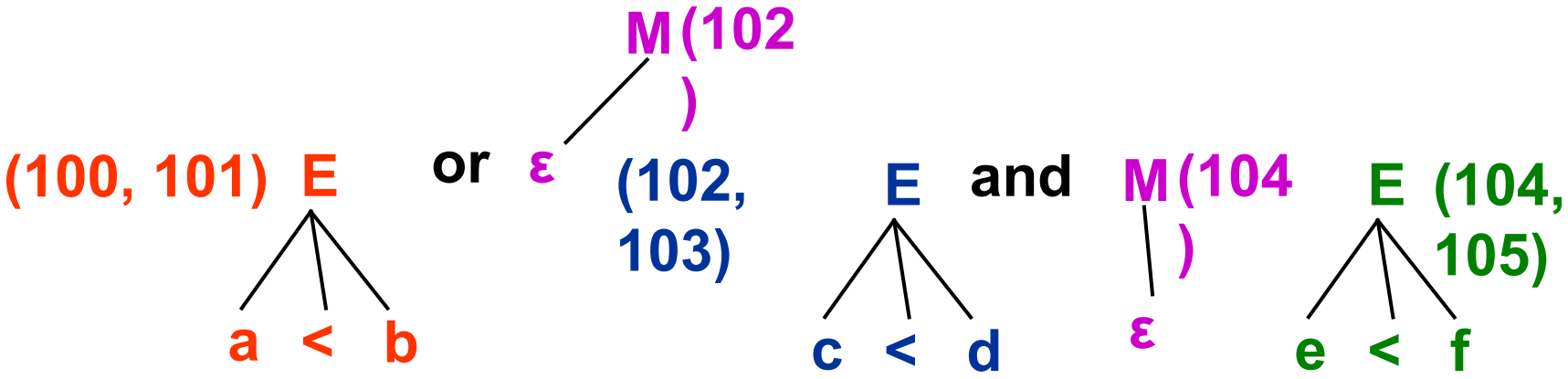
(2) $E \rightarrow E_1$ and M E_2

{ backpatch(E_1.truelist, M.quad);
E.truelist:=E_2.truelist;
E.falselist:=merge(E$_1$.falselist,E$_2$.falselist) }

a<b or c<d and e<f

(5) $E \rightarrow id_1$ 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') }
(7) M→ε { M.quad:=nextquad }

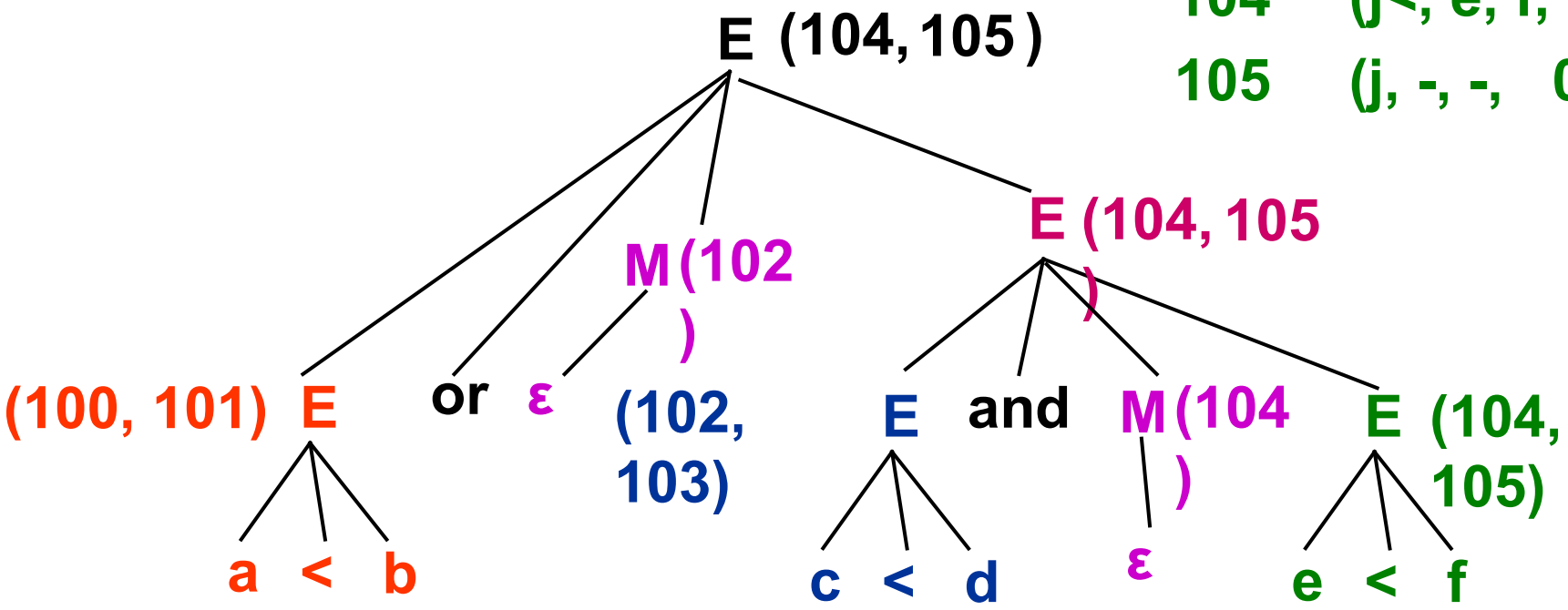
100	(j<, a, b, 0)
101	(j, -, -, 0)
102	(j<, c, d, 0)
103	(j, -, -, 0)
104	(j<, e, f, 0)
105	(j, -, -, 0)



(1) $E \rightarrow E_1$ or M E_2
{ backpatch(E_1.falselist, M.quad);
$E.truelist := merge(E_1.truelist, E_2.truelist)$;
$E.falselist := E_2.falselist$ }

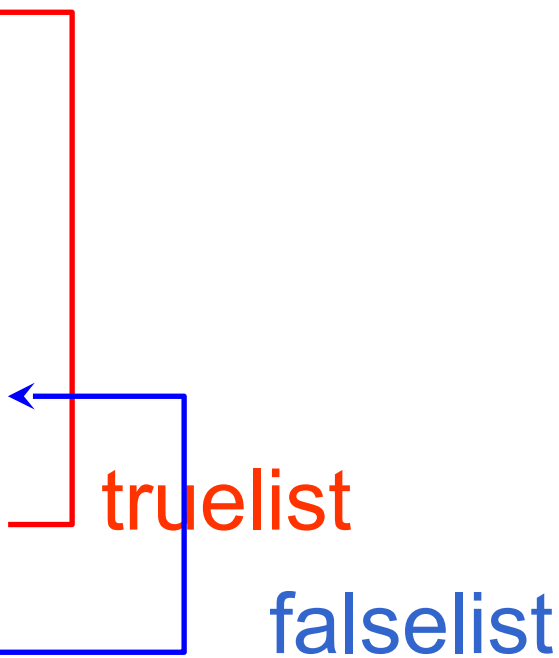
(2) $E \rightarrow E_1$ and M E_2
{ backpatch(E_1.truelist, M.quad);
$E.truelist := E_2.truelist$;
$E.falselist := merge(E_1.falselist, E_2.falselist)$ }

100 (j<, a, b, 0) 102
101 (j, -, -, 0) 104
102 (j<, c, d, 0) 100
103 (j, -, -, 103)
104 (j<, e, f, 0)
105 (j, -, -, 0)



a<b or c<d and e<f

100 (j<, a, b, 0)
101 (j, -, -, 102)
102 (j<, c, d, 104)
103 (j, -, -, 0)
104 (j<, e, f, 100)
105 (j, -, -, 103)



小结

- 布尔表达式的翻译
  - 数值表示法
  - 作为条件控制的布尔式翻译
    - 一遍扫描的翻译模式

作业

- P218-6