SNS COLLEGE OF TECHNOLOGY

(Autonomous)

COIMBATORE-35

ICG- Backpatching

Backpatching

The syntax directed definition can be implemented in two or more passes (we have both synthesized attributes and inheritent attributes).

- Build the tree first
- Walk the tree in the depth first order.

The main difficulty with code generation in **one** pass is that we may not know the target of a branch when we generate code for flow-of –control statements

Backpatching is the technique to get around this problem.

- Generate branch instructions with empty targets
- When the target is known, fill in the label of the branch instructions (backpatching).

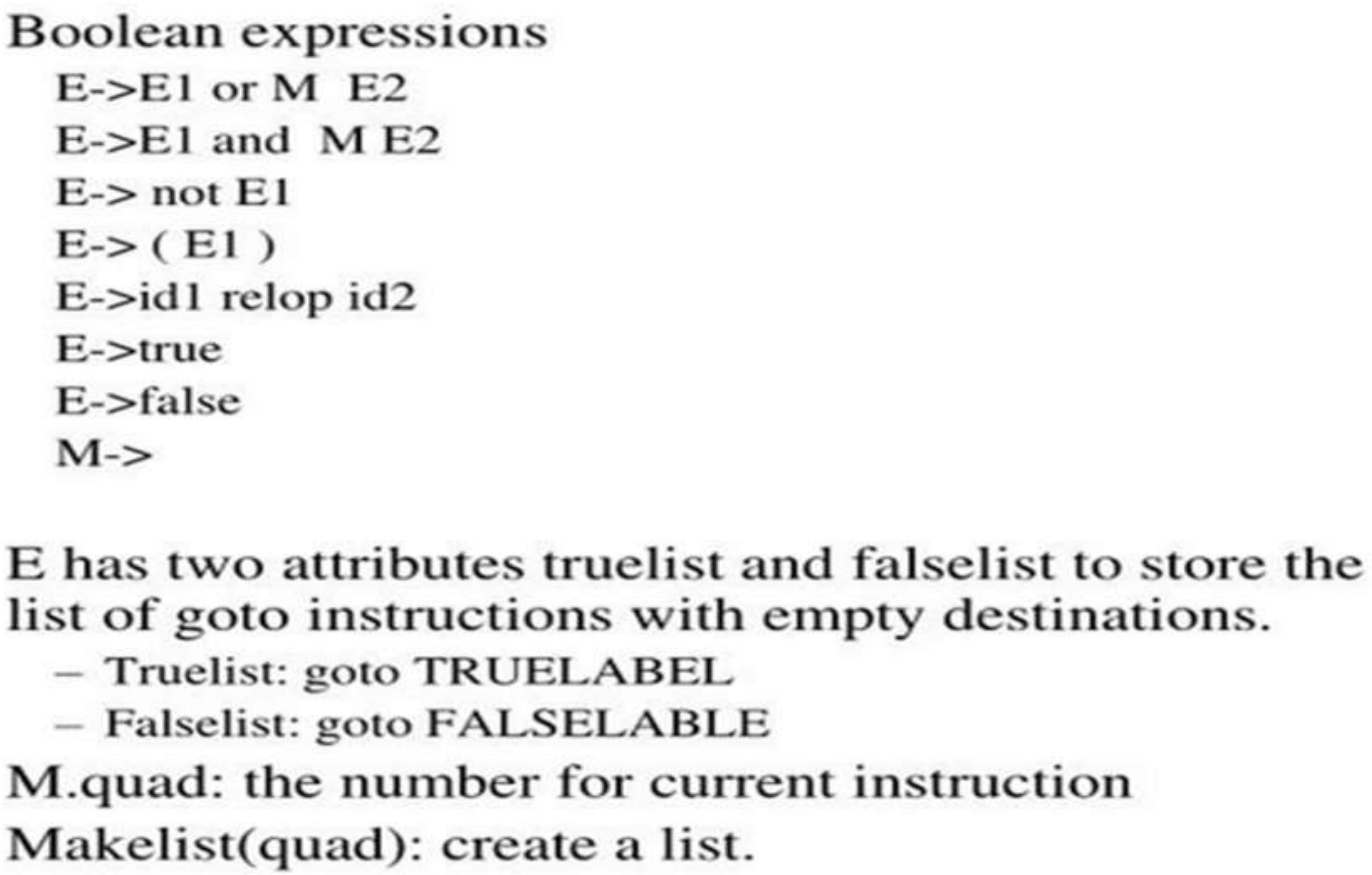
Backpatching

Example: generate intermediate code for the following program segment

If (a < b) then I := I+1 else j:= I+1

```
100:   if a < b then goto ???
101:   goto ???
102:   t1 = I+1
103:   I = t1
104:    goto ???
105:   t1 = I+1
106:   j = t1
107:
```

Backpatching



Boolean expressions

- E->E1 or M E2
- E->E1 and M E2
- E-> not E1
- E-> (E1)
- E->id1 relop id2
- E->true
- E->false
- M->

E has two attributes truelist and falselist to store the list of goto instructions with empty destinations.

- Truelist: goto TRUELABEL
- Falselist: goto FALSELABLE

M.quad: the number for current instruction

Makelist(quad): create a list.

Backpatching

```
E->E1 or M  E2 { backpatch(E1.falselist, M.qual);
     E.truelist = merge(E1.truelist, E2.truelist); E.falselist = E2.falselist;}
E->E1 and  M E2 {backpatch(E1.truelist, M.qual);
     E.truelist = E2.truelist; E.falselist = merge(E1.truelist, E2.truelist);}
E-> not E1 {E.truelist = E1.falselist, E.falselist = E1.truelist;}
E-> ( E1 )  {E.truelist = E1.truelist; E.falselist = E1.falselist;}
E->id1 relop id2 {E.truelist = makelist(nextquad);
     E.falselist = makelist(nextquad+1);
     emit(‘if’ id1.place  relop.op   id2.place ‘goto ???’);
     emit(‘goto  ???’);
}
E->true  {E.truelist = makelist(nextquad); emit(‘goto ???’);}
E->false {E.falselist = makelist(nextquad);emit(‘goto ???’);}
M->     {M.quad = nextquad;}
```

Backpatching

Example: generating code using one pass LR parser

a < b or (c < d and e < f)

```
100:  If  (a < b) goto ???
101:  goto ???
102:  if c < d goto ???
103: goto ???
104:  If e < f goto ???
105:  goto ???
```

Backpatching

Flow of control statements

S -> if E then S1

S -> if E then S1 else S2

S -> while E do S1

S-> begin L end

S-> A

L->L; S

L->S

- Attributes: E.truelist, E.falselist
- S.nextlist: goto statements to the next instruction after this statement.

Backpatching

S -> if E then M S1 { backpatch(E.truelist, M.quad);

S.nextlist = merge(E.falselist, S1.nextlist);}

S-> if E then M1 S1 N else M2 S2 {

backpatch(E.truelist, M1.quad); backpatch(E.falselist, M2.quad);

S.nextlist := merge(S1.nextlist, N.nextlist, S2.nextlist);}

S -> while M1 E do M2 S1 {backpatch(E.truelist, M2.quad);

Backpatch(S1.nextlis alselist;

Emit('goto ' M1.qua

S-> begin L end {S.

S-> A {S.nextlist =

L->L; M S {backpat xtlist = S.nextlist;}

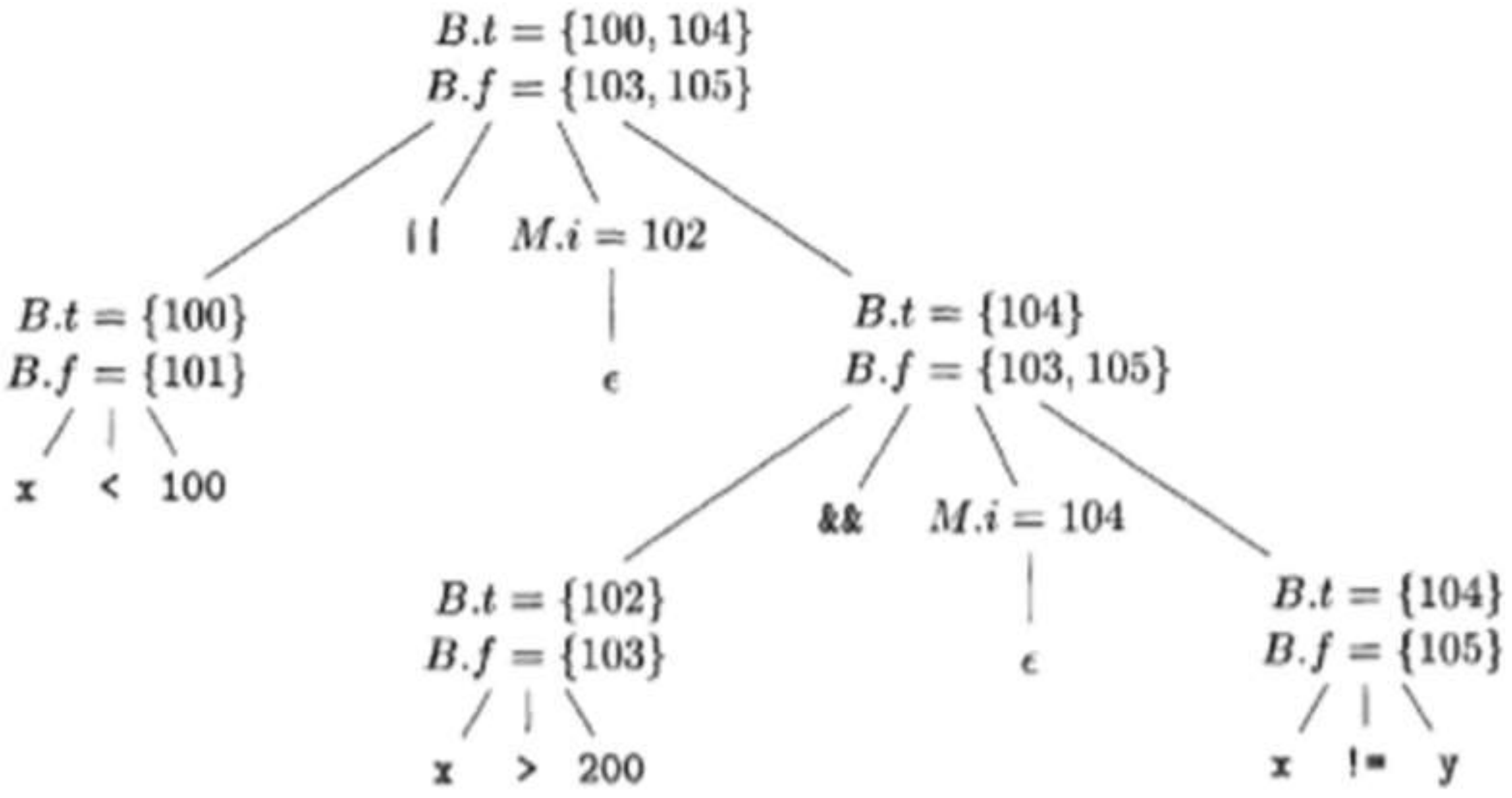
L->S {L.nextlist = S.nextlist;}

N-> {N.nextlist = makelist(nextquad); emit('goto ???');}

M-> {M.quad = nextquad;}

```
While  (a < b or (c < d and e < f)) do
 if c < d then
   x:= y+z
 else
   x:= y-z
```

Backpatching



Summarization