for-loop: translation into intermediate code

for-loop

<u>Syntax:</u>

S $\rightarrow$ for (S1; E; S2) S3

for-loop

<u>Syntax:</u>

S → for (S1; E; S2) S3

<u>Roles</u>:

for-loop

Syntax:

$S \rightarrow$ for (S1; E; S2) S3

Roles:

S1 – data initialization

for-loop

<u>Syntax:</u>

$S \rightarrow$ for (S1; E; S2) S3

<u>Roles</u>:

S1 – data initialization

E – iteration condition

for-loop

Syntax:

$S \rightarrow$ for (S1; E; S2) S3

Roles:

S1 – data initialization

E – iteration condition

S2 – data update

for-loop

Syntax:

S $\rightarrow$ for (S1; E; S2) S3

Roles:

S1 – data initialization

E – iteration condition

S2 – data update

S3 – body of the loop

for-loop

Syntax:

$S \rightarrow$ for (S1; E; S2) S3

Roles:

S1 – data initialization

E – iteration condition

S2 – data update

S3 – body of the loop

Example:

```
for (if k = 0 then j:=0 else j:=10; j < 100; j:=j + 5)  m := m*j
```

for-loop

<u>Semantics</u>:

```
1.  perform initialization S1
2.  if E do
        S3  /* execute the loop body */
        S2  /* update data  */
        return to 2
3.  next statement after the for-loop
```

for-loop : structure of the intermediate code

S → for (S1; E; S2) S3

$\Downarrow$

```
              S1.code
loop_start:   E.code
              if (E.place = 0) goto S.next
              S3.code
              S2.code
              goto loop_start
S.next:       ......................
```

NOTE: S2 is derived **before** S3, but S2.code is executed **after** S3.code

Reminder: inherited attribute S.next

- Translation of S :

  need to know the label for the next (after S) statement in the translated program

- Name of this label is the value of S.next

  - It should be known **in advance**, before derivation from S

- Inheritance starts at the program level (at the derivation rule for the initial variable P)

Translation scheme	Intermediate code
$P \rightarrow D$ {S.next := newlabel} S	S.code
	S.next: /* end of program */

Updated structure of the intermediate code

```
            S1.code
S1.next:    E.code                          /* S1.next is loop_start */
            if (E.place = 0) goto S.next
            S3.code                         תמונת ה " פאזל" שצריך להרכיב
S3.next:    S2.code
S2.next:    goto S1.next
S.next:     ......................
```

<u>NOTE</u>:

- the name of label Si.next (i = 1,2,3) should be known **before** derivation from Si (because it is used in creation of intermediate code for Si)

Syntax-directed translation into intermediate code

```
S → for (
          { S1.next = newlebel }                /*  label for loop_start  */
     S1                                         /*  parse_S (S1.next)  returns  S1.code  */
      ;
     E                                          /*  parse_E()  returns  E.code  and  E.place  */
      ;
          { S2.next = newlabel }
     S2                                         /*  parse_S (S2.next)  returns  S2.code  */
      )
          { S3.next = newlabel }
     S3                                         /*  parse_S (S3.next)  returns  S3.code  */
        { S.code =
          S1.code  ||
          S1.next  ||  ’:’  ||  E.code  ||
          'if' || E.place  ||  '=0 goto'  ||  S.next ||
          S3.code  ||
          S3.next || ‘:’ || S2.code ||
          S2.next  ||  ‘: goto’  ||  S1.next  ||
          S.next  ||  ‘:’
  }
```

יצירה של

"חתיכות פאזל"

הרכבת ה "פאזל"

Syntax-directed translation into intermediate code

```
S → for (
            { S1.next = newlebel }                  /*  label for loop_start  */
     S1                                             /*  parse_S (S1.next)  returns  S1.code  */
      ;
     E                                              /*  parse_E()  returns  E.code  and  E.place  */
      ;
            { S2.next = new_label }
     S2                                             /*  parse_S (S1.next)  returns  S2.code  */
      )
            { S3.next = newlabel }
     S3                                             /*  parse_S (S3.next)  returns  S3.code  */
        { S.code =
            S1.code  ||
            S1.next  ||  ':'  ||  E.code  ||
            'if' || E.place  ||  '=0 goto'  ||  S.next ||
            S3.code  ||
            S3.next || ':' || S2.code ||
            S2.next  ||  ': goto'  ||  S1.next  ||
            S.next  ||  ':'
   }
```

יצירה של

"חתיכות פאזל"

הרכבת ה "פאזל"

Syntax-directed translation into intermediate code

```
S → for (
          { S1.next = newlebel }                /*  label for loop_start  */
     S1                                         /*  parse_S (S1.next)  returns  S1.code  */
      ;
     E                                          /*  parse_E()  returns  E.code  and  E.place  */
      ;
          { S2.next = new_label }
     S2                                         /*  parse_S (S1.next)  returns  S2.code  */
      )
          { S3.next = newlabel }
     S3                                         /*  parse_S (S3.next)  returns  S3.code  */
       { S.code =
          S1.code  ||
          S1.next  ||  ':'  ||  E.code  ||
          'if' || E.place  ||  '=0 goto'  ||  S.next ||
          S3.code  ||
          S3.next || ':' || S2.code ||
          S2.next  ||  ': goto'  ||  S1.next  ||
          S.next  ||  ':'
  }
```

יצירה של

"חתיכות פאזל"

הרכבת ה "פאזל"

Syntax-directed translation into intermediate code

```
S → for (
          { S1.next = newlebel }              /*  label for loop_start  */
     S1                                       /*  parse_S (S1.next)  returns  S1.code  */
      ;
     E                                        /*  parse_E()  returns  E.code  and  E.place  */
      ;
          { S2.next = new_label }
     S2                                       /*  parse_S (S1.next)  returns  S2.code  */
      )
          { S3.next = newlabel }
     S3                                       /*  parse_S (S3.next)  returns  S3.code  */
        { S.code =
          S1.code  ||
          S1.next  ||  ’:’  ||  E.code  ||
          'if' || E.place  ||  '=0 goto'  ||  S.next ||
          S3.code  ||
          S3.next || ‘:’ || S2.code ||
          S2.next  ||  ‘: goto’  ||  S1.next  ||
          S.next  ||  ‘:’
  }
```

יצירה של "חתיכות פאזל"

הרכבת ה "פאזל"

Syntax-directed translation into intermediate code

```
S → for (
          { S1.next = newlebel }               /*  label for loop_start  */
    S1                                         /*  parse_S (S1.next)  returns  S1.code  */
    ;
    E                                          /*  parse_E()  returns  E.code  and  E.place  */
    ;
          { S2.next = new_label }
    S2                                         /*  parse_S (S1.next)  returns  S2.code  */
     )
          { S3.next = newlabel }
    S3                                         /*  parse_S (S3.next)  returns  S3.code  */
      { S.code =
          S1.code  ||
          S1.next  ||  ’:’  ||  E.code  ||
          'if' || E.place  ||  '=0 goto'  ||  S.next ||
          S3.code  ||
          S3.next || ‘:’ || S2.code ||
          S2.next  ||  ‘: goto’  ||  S1.next  ||
          S.next  ||  ‘:’
  }
```

יצירה של

"חתיכות פאזל"

הרכבת ה "פאזל"

Syntax-directed translation into intermediate code

```
S → for (
          { S1.next = newlebel }          /*  label for loop_start  */
     S1                                   /*  parse_S (S1.next)  returns  S1.code  */
      ;
     E                                    /*  parse_E()  returns  E.code  and  E.place  */
      ;
          { S2.next = new_label }
     S2                                   /*  parse_S (S1.next)  returns  S2.code  */
      )
          { S3.next = newlabel }
     S3                                   /*  parse_S (S3.next)  returns  S3.code  */
       { S.code =
          S1.code  ||
          S1.next  ||  ':'  ||  E.code  ||
          'if' || E.place  ||  '=0 goto'  ||  S.next ||
          S3.code  ||
          S3.next || ':' || S2.code ||
          S2.next  ||  ': goto'  ||  S1.next  ||
          S.next  ||  ':'
  }
```

יצירה של

"חתיכות פאזל"

הרכבת ה "פאזל"

Syntax-directed translation into intermediate code

```
S → for (
            { S1.next = newlebel }                /*  label for loop_start  */
     S1                                           /*  parse_S (S1.next)  returns  S1.code  */
      ;
     E                                            /*  parse_E()  returns  E.code  and  E.place  */
      ;
            { S2.next = new_label }
     S2                                           /*  parse_S (S1.next)  returns  S2.code  */
      )
            { S3.next = newlabel }
     S3                                           /*  parse_S (S3.next)  returns  S3.code  */
         { S.code =
            S1.code  ||
            S1.next  ||  ':'  ||  E.code  ||
            'if' || E.place  ||  '=0 goto'  ||  S.next ||
            S3.code  ||
            S3.next || ':' || S2.code ||
            S2.next  ||  ': goto'  ||  S1.next  ||
            S.next  ||  ':'
   }
```

יצירה של

"חתיכות פאזל"

הרכבת ה "פאזל"

Example

S $\rightarrow^*$ for (j:=1; j ≤ 100; j:= j*2) k := k+j ;

```
                                  /*  S.next = “L1”   */
    j:=1                          /*  S1.code
                                      S1.next = “L2”  */
L2: if j ≤ 100 goto L3
    t1 := 0                       /* E.code
    goto L4                          E.place = “t1”  */
L3: t1 := 1

L4: if t1 = 0 goto L1
    t2 := k + j                   /*  S3.code
    k := t2                           S3.next = “L6”  */

L6: t3 := j*2                     /*  S2.code
    j := t3                           S2.next = “L5”  */
L5: goto L1
L1:                               /*  S.next = “L1”  */
```

```
              S1.code

S1.next:   E.code

              If (E.place == 0) goto S.next
              S3.code

S3.next:   S2.code

S2.next:   goto  S1.next
S.next:
```