Δομές Ενδιάμεσου Κώδικα

Εκπαιδευτικές Δομές

Διαλέξεις στο μάθημα: Μεταφραστές
Γεώργιος Μανής

Η δομή until_equal

until_equal(exp^1,exp^2)

< : S^1

> : S^2

== : S^3

- Αποτιμούνται τα *exp*^{1} και *exp*^{2}.
  Αν *exp*^{1} < *exp*^{2} τότε εκτελούνται τα S^1 και ο έλεγχος μεταβαίνει στην αρχή της δομής.
  Αν *exp*^{1} > *exp*^{2} τότε εκτελούνται τα S^2 και ο έλεγχος μεταβαίνει στην αρχή της δομής.
  Τέλος, αν *exp*^{1} == *exp*^{2} τότε εκτελούνται τα S^3 *και ο έλεγχος βγαίνει έξω από τη δομή.*

Η δομή until_equal

```
until_equal
 (exp1,exp2)
  <  : S1
  >  : S2
 ==  : S3
```

Η δομή until_equal

p0: condquad=nextquad()

until_equal

{p0} (exp^1,exp^2)

< : S^1

> : S^2

== : S^3

Η δομή until_equal

until_equal

{p0} (exp^1,exp^2) **{p0’}**

  < : S^1

  > : S^2

 == : S^3

```
p0:  condquad=nextquad()
p0': smaller_list = makelist(nextquad)
     genquad(“<",exp1.place,exp2.place,"_")
     larger_list = makelist(nextquad)
     genquad(">",exp1.place,exp2.place,"_")
     equal_list = makelist(nextquad)
     genquad(“=",exp1.place,exp2.place,"_")
```

Η δομή until_equal

until_equal

{p0} (exp^1,exp^2) **{p0’}**

< **{p1}** : S^1

> : S^2

== : S^3

```
p0:  condquad=nextquad()
p0': smaller_list = makelist(nextquad)
     genquad(“<",exp1.place,exp2.place,"_")
     larger_list = makelist(nextquad)
     genquad(">",exp1.place,exp2.place,"_")
     equal_list = makelist(nextquad)
     genquad(“=",exp1.place,exp2.place,"_")

p1:  backpatch(smaller_list,nextquad())
```

Η δομή until_equal

until_equal

{p0} (exp[1],exp[2]) **{p0’}**

  < **{p1}** : S[1] **{p1’}**

  > : S[2]

 == : S[3]

```
p0:  condquad=nextquad()
p0': smaller_list = makelist(nextquad)
     genquad(“<",exp1.place,exp2.place,"_")
     larger_list = makelist(nextquad)
     genquad(">",exp1.place,exp2.place,"_")
     equal_list = makelist(nextquad)
     genquad(“=",exp1.place,exp2.place,"_")

p1:  backpatch(smaller_list,nextquad())
p1': genquad("jmp","_","_",condquad)
```

Η δομή until_equal

until_equal

{p0} (exp[1],exp[2]) **{p0’}**

  < **{p1}** : S[1] **{p1’}**

  > **{p2}** : S[2]

 == : S[3]

```
p0:  condquad=nextquad()
p0': smaller_list = makelist(nextquad)
     genquad(“<",exp1.place,exp2.place,"_")
     larger_list = makelist(nextquad)
     genquad(">",exp1.place,exp2.place,"_")
     equal_list = makelist(nextquad)
     genquad(“=",exp1.place,exp2.place,"_")

p1:  backpatch(smaller_list,nextquad())
p1': genquad("jmp","_","_",condquad)

p2:  backpatch(larger_list,nextquad())
```

Η δομή until_equal

until_equal

{p0} (exp^1,exp^2) **{p0’}**

  < **{p1}** : S^1 **{p1’}**

  > **{p2}** : S^2 **{p2’}**

 == : S^3

```
p0:  condquad=nextquad()
p0’: smaller_list = makelist(nextquad)
     genquad(“<",exp1.place,exp2.place,"_")
     larger_list = makelist(nextquad)
     genquad(">",exp1.place,exp2.place,"_")
     equal_list = makelist(nextquad)
     genquad(“=",exp1.place,exp2.place,"_")

p1:  backpatch(smaller_list,nextquad())
p1': genquad("jmp","_","_",condquad)

p2:  backpatch(larger_list,nextquad())
p2': genquad("jmp","_","_",condquad)
```

Η δομή until_equal

until_equal

{p0} (exp[1],exp[2]) **{p0’}**

< **{p1}** : S[1] **{p1’}**

> **{p2}** : S[2] **{p2’}**

== **{p3}** : S[3]

```
p0:  condquad=nextquad()
p0’: smaller_list = makelist(nextquad)
     genquad(“<",exp1.place,exp2.place,"_")
     larger_list = makelist(nextquad)
     genquad(">",exp1.place,exp2.place,"_")
     equal_list = makelist(nextquad)
     genquad(“=",exp1.place,exp2.place,"_")

p1:  backpatch(smaller_list,nextquad())
p1': genquad("jmp","_","_",condquad)

p2:  backpatch(larger_list,nextquad())
p2': genquad("jmp","_","_",condquad)

p3:  backpatch(equal_list,nextquad())
```

Η δομή incase

incase (**(**cond**)** S^1)*

default: S^2

- Ελέγχονται μία προς μία οι συνθήκες *cond* . Για κάθε συνθήκη που ισχύει εκτελείται το αντίστοιχο S^1 και μετά ο έλεγχος μεταβαίνει στον επόμενο έλεγχο συνθήκης αν υπάρχει. Όταν ελεγχθούν όλες οι cond, τότε εάν έστω και ένα από τα S^1 έχει εκτελεστεί, ο έλεγχος μεταβαίνει στην αρχή της **incase**. Εάν κανένα από τα S^1 δεν έχει εκτελεστεί, τότε εκτελούνται οι εντολές μέσα στο S^2 και στη συνέχεια ο έλεγχος μεταβαίνει έξω από την **incase**.

Η δομή incase

incase (**(**cond**)** S^1)*

default: S^2

Η δομή incase

incase **{p0}** (**(**cond**)** S^1)*

default: S^2

p0: t=newTemp()
first_quad= nextquad()
genquad (‘:=’, ‘0’, ‘_’, t)

Η δομή incase

incase **{p0}** (**(**cond**)** **{p1}** S^1)*

default: S^2

```
p0:     t=newTemp()
        first_quad = nextquad()
        genquad (‘:=’, ‘0’, ‘_’, t)
p1:     backpatch (cond.true, nextquad())
```

Η δομή incase

incase **{p0}** (**(**cond**)** **{p1}** S^1 **{p2}**)*

default: S^2

```
p0:     t=newTemp()
        first_quad = nextquad()
        genquad (‘:=’, ‘0’, ‘_’, t)
p1:     backpatch (cond.true, nextquad())
p2:     genquad(‘:=’, ‘1’, ‘_’, t)
        backpatch(cond.false, nextquad())
```

Η δομή incase

incase **{p0}** (**(**cond**)** **{p1}** S^1 **{p2}**)*

default: **{p3}** S^2

```
p0:     t=newTemp()
        first_quad = nextquad()
        genquad (‘:=’, ‘0’, ‘_’, t)
p1:     backpatch (cond.true, nextquad())
p2:     genquad(‘:=’, ‘1’, ‘_’, t)
        backpatch(cond.false, nextquad())
p3:     genquad(‘=’, ‘1’, ‘t’ first_quad)
```

Η δομή double_while

```
double_while (cond) S1
        else S2
```

- Την πρώτη φορά που ο έλεγχος εισέρχεται στον βρόχο, αποφασίζει μέσα από την *cond* αν θα πάει στ0 S^1 (*true*) ή στο S^2 (*false*). Από το S^1 φεύγει, όταν η συνθήκη σταματήσει να είναι *true*. Από το S^2 φεύγει όταν η συνθήκη σταματήσει να είναι *false*. Και στις δύο αυτές περιπτώσεις ο έλεγχος φεύγει έξω από τον βρόχο. Δηλαδή, δεν είναι ποτέ δυνατόν σε μία εκτέλεση της **double_while** ο έλεγχος να περάσει και από την S^1 και από την S^2 .

Η δομή double_while

double_while

(cond) S^1

else S^2

Η δομή double_while

```
p0:   state = newTemp()
      genquad(“:=”, “0”, “”, state)
      condquad = nextquad()
```

double_while **{p0}**

(cond) S^1

else S^2

Η δομή double_while

```
double_while {p0}

    (cond) {p1} S1

    else S2
```

```
p0:   state = newTemp()
      genquad(“:=”, “0”, “”, state)
      condquad = nextquad()

p1:   backpatch(cond.true, nextquad())
      state1_list = makelist(nextquad)
      genquad(“=”, “2”, state, “_”)
      genquad(“:=”, “1”, “”, state)
```

Η δομή double_while

```
double_while {p0}

    (cond) {p1} S1 {p1'}

    else S2
```

```
p0:   state = newTemp()
      genquad(“:=”, “0”, “”, state)
      condquad = nextquad()

p1:   backpatch(cond.true, nextquad())
      state1_list = makelist(nextquad)
      genquad(“=”, “2”, state, “_”)
      genquad(“:=”, “1”, “”, state)
p1':  genquad(“jmp”, “”, “”,condquad)
```

Η δομή double_while

```
double_while {p0}

    (cond) {p1} S1 {p1'}

    else {p2} S2
```

```
p0:   state = newTemp()
      genquad(“:=”, “0”, “”, state)
      condquad = nextquad()

p1:   backpatch(cond.true, nextquad())
      state1_list = makelist(nextquad)
      genquad(“=”, “2”, state, “_”)
      genquad(“:=”, “1”, “”, state)
p1':  genquad(“jmp”, “”, “”,condquad)

p2:   backpatch(cond.false, nextquad())
      state2_list = makelist(nextquad)
      genquad(“=”, “1”, state, “_”)
      genquad(“:=”, “2”, “”, state)
```

Η δομή double_while

```
double_while {p0}

    (cond) {p1} S1 {p1'}

    else {p2} S2 {p2‘}
```

```
p0:   state = newTemp()
      genquad(“:=”, “0”, “”, state)
      condquad = nextquad()

p1:   backpatch(cond.true, nextquad())
      state1_list = makelist(nextquad)
      genquad(“=”, “2”, state, “_”)
      genquad(“:=”, “1”, “”, state)
p1':  genquad(“jmp”, “”, “”,condquad)

p2:   backpatch(cond.false, nextquad())
      state2_list = makelist(nextquad)
      genquad(“=”, “1”, state, “_”)
      genquad(“:=”, “2”, “”, state)
p2':  genquad(“jmp”, “”, “”,condquad)
```

Η δομή double_while

```
double_while {p0}

    (cond) {p1} S1 {p1'}

    else {p2} S2 {p2'} {p3}
```

```
p0:   state = newTemp()
      genquad(“:=”, “0”, “”, state)
      condquad = nextquad()

p1:   backpatch(cond.true, nextquad())
      state1_list = makelist(nextquad)
      genquad(“=”, “2”, state, “_”)
      genquad(“:=”, “1”, “”, state)
p1':  genquad(“jmp”, “”, “”,condquad)

p2:   backpatch(cond.false, nextquad())
      state2_list = makelist(nextquad)
      genquad(“=”, “1”, state, “_”)
      genquad(“:=”, “2”, “”, state)
p2':  genquad(“jmp”, “”, “”,condquad)

p3:   backpatch(state1_list,nextquad())
      backpatch(state2_list,nextquad())
```

Ευχαριστώ