Recursive Programming in Lisp L2

Robert Krisztian Sandor, group 927

10.12.2015

10. Decide whether a tree of type (2) is balanced (the difference between the depths of the two subtrees is not larger than 1).
(A (B) (C (D) (E))) (2)

Mathematical Models

$$maxlevel(l_1, ..., l_n) = \begin{cases} 0 & if\ n = 0 \\ 1 & if\ n = 1 \\ 1 + maxchildren(l_3, ..., l_n, maxlevel(l_2)) & otherwise \end{cases}$$

$$maxchildren(l_1, ..., l_n, cmax) = \begin{cases} cmax & if\ n = 0 \\ maxchildren(l_2, ..., l_n, max(cmax, maxlevel(l_2))) & otherwise \end{cases}$$

$$minlevel(l_1, ..., l_n) = \begin{cases} 0 & if\ n = 0 \\ 1 & if\ n = 1 \\ 1 + minchildren(l_3, ..., l_n, minlevel(l_2)) & otherwise \end{cases}$$

$$minchildren(l_1, ..., l_n, cmin) = \begin{cases} cmin & if\ n = 0 \\ minchildren(l_2, ..., l_n, min(cmin, minlevel(l_2))) & otherwise \end{cases}$$

$$balanced(l_1, ..., l_n) =\mid maxlevel(l_1, ..., l_n) - minlevel(l_1, ..., l_n) \mid <= 1$$

Meaning of predicates. Flow models. Source Code

```
; maxlevel(tree : List)
; tree - list representing a tree of type (2)
(defun maxlevel (tree)
  (cond
    ((null tree) 0)
    ((null (cdr tree)) 1)
    (t (1+ (maxchildren (cddr tree) (maxlevel (cadr tree)))))
    )
  )
; maxchildren(tree : List, cmax : Integer)
; tree - a child tree of type (2)
; cmax - current max depth
(defun maxchildren (children cmax)
  (cond
    ((null children) cmax)
    (t (maxchildren (cdr children) (max cmax (maxlevel (car children)))))
    )
  )

; minlevel(tree : List)
; tree - list representing a tree of type (2)
(defun minlevel (tree)
  (cond
    ((null tree) 0)
    ((null (cdr tree)) 1)
    (t (1+ (minchildren (cddr tree) (minlevel (cadr tree)))))
    )
  )
; minchildren(tree : List, cmin : Integer)
; tree - a child tree of type (2)
; cmin - current min depth
(defun minchildren (children cmin)
  (cond
    ((null children) cmin)
    (t (minchildren (cdr children) (min cmin (minlevel (car children)))))
    )
  )
```

```
; balanced(tree : List)
; tree - list representing a tree of type (2)
(defun balanced (tree)
  (<= (abs (- (maxlevel tree) (minlevel tree))) 1)
  )
```

Examples

```
[5]> (balanced '(A (B (D) (E)) (C)))
T

[6]> (balanced '(A (B) (C(D(E(F))))))
NIL
```