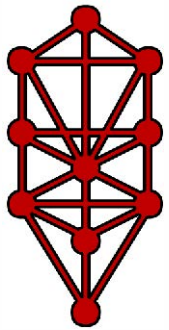
This page is intentionally left blank.

Kazimir Majorinc

Funkcija funkcije

FUNCTION

Povijest Lispa 46.



Razmjena vještina
Hacklab u mami
25. siječnja 2014.

JOEL MOSES, *The Function of FUNCTION in LISP or Why the FUNARG Problem Should be Called the Environment Problem,* MIT AI-199. June 1970, 15 str.

Memo govori o sad već dobro poznatom FUNARG problemu. Neki dijelovi memoa su jednostavni i o njima smo već puno puta govorili. Neki dijelovi su nejasni, jer govore o implementacijskim detaljima koji nisu opisani u ranijim dokumentima. Najvažnije je razumljivo: otkriće tzv. „full environment problema“, kasnije „upward funarg problema“. (**Moses** ne koristi taj pojam).

okolina aktivacije
(activation environment)

okolina [u]vezivanja)
(binding environment)

```
define f(x) =
  begin;
  if a = 0 then return x
           else return -x;
     end;

define g(x,fun) =
  begin local a;
  a = 0
  return x + fun(x);
  end;

begin local a,b
  a = 1
  b = g(3,f);
end;
```

Taj problem je riješen u svim implementacijama. Međutim,

```
define f(x) =
  begin;
  if a = 0 then return x
           else return -x;
end;
define g(x) =
  begin local a;
  a = 2;
  return f
end;
begin local a,b,h;
  a = 0;
  h = g(2);
  b = h(3);
end;
```

okolina vezivanja

okolina aktivacije

U Lispu, funkcije se pozivaju ili sa `(FUNCTION f)` ili sa `(QUOTE f)`. Mnemonički trik je da se poziv `(QUOTE f)` smatra otvorenom funkcijom, funkcijom iz koje su „pobjegle“ slobodne varijable.

```
(SETQ A 3)
(DO-SOMETHING (QUOTE (LAMBDA(X)(+ X A))))
(DO-SOMETHING (FUNCTION (LAMBDA (X) (+ X A))))
```

Sistem mora pamtiti sve okoline u kojima je f definiran.

Programeri rijetko kad vraćaju zatvorene lambda-izraze, pa su rijetko i dolazili u situaciju u kojoj bi naišli na „full environment“ problem. Samo interpreteri koji uopće nisu koristi optimizacije (nego su vrijednosti varijabli spremane u alisti) funkcioniraju dobro.

kraj