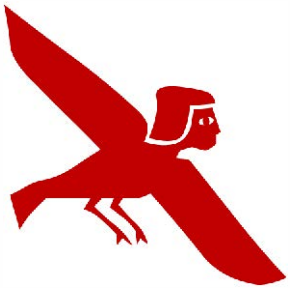
This page is intentionally left blank.

Kazimir Majorinc

DEAN E. WOOLDRIDGE

Povijest Lispa 34.



Razmjena vještina
Hacklab u mami
28. rujna 2013.

1963. Dean E. Wooldridge, An algebraic simplify program in LISP

```
if eq[caar[r];TIMES] then
   if numberp[cadr[car[r]]] then
     collect[caddr[car[r]];times[cadar[r];mm];tm]
       (if a numeric factor other than one is involved in a
       product, then simptimes places this factor at the
       front of the list of factors; thus, by the time
       simplus sees the product, if a numeric factor (not 1)
       appears, then it is necessarily the first factor;
       similar remarks may be made with respect to simplus
       and non-zero numeric terms);
 otherwise collect[car[r];mm;tm];
otherwise collect[car[r];mm;tm].);
```

1964. Dean E. Wooldridge, The new LISP 1.55. system

Unapređenja na Stanfordu u odnosu na LISP 1.5, redom nebitna. Interpreter se zaustavlja kad dođe do greške, dok je LISP 1.5 javio grešku i nastavio …

LISP 1.5 je ispisivao „The time has come, the walrus said …“ LISP 1.55 to više ne ispisuje.

Ispisivanje i čitanje sa bušenih traka.

1965. McCarthy & Wooldridge, A solution of the functional arguments problem in LISP.

Primjerice, neka je funkcija `TESTR` definirana lambda-izrazom

```
(LAMBDA (L FN)
        (COND ((NULL L) NIL)
              (T        (TESTR (CAR L)
                               (QUOTE (LAMBDA ()
                                              (CONS (CDR L)
                                                    (FN))))))))
```

Nakon poziva

```
(TESTR (QUOTE ((A) (B) (C)))
       (QUOTE (LAMBDA (X) X)))
```

parametri `L` i `FN` bi imali slijedeće vrijednosti:

`L`: `((A) (B) (C))`
`FN`: `(LAMBDA (X) X)`.

Kako nije zadovoljen uvjet `(NULL L)`, izvršavala bi se druga grana *cond-izraza*

```
(TESTR (CAR L)
       (QUOTE (LAMBDA ()
                      (CONS (CDR L)
                            (FN)))))
```

Nakon ponovnog poziva funkcije **TESTR**, vrijednosti parametara bi bile

L: **(A)**
FN: **(LAMBDA()(CONS (CDR L) (FN)))**.

U izrazu koji je vrijednost parametra **FN** pojavljuju se simboli **L** i **FN** , ali vrijednost tih simbola je sada različita od vrijednosti koju su imali u trenutku u kojem je pozvana funkcija **TESTR**.

Da bi LISP interpreter korektno izračunavao ovakve izraze, nužno je da zajedno sa izrazima budu sačuvane i originalne vrijednosti varijabli. U LISPu 1.5 za to služi operator **FUNCTION** posredstvom kojeg se funkcije mogu davati kao argumenti drugih funkcija. Tako, **TESTR** treba definirati kao

```
(LAMBDA (L FN)
  (COND ((NULL L) NIL)
        (T        (TESTR (CAR L)
                         (FUNCTION (LAMBDA ()
                                           (CONS (CDR L)
                                                 (FN))))))))).
```

Tada će u gornjem primjeru, nakon poziva

```
(TESTR (QUOTE ((A) (B) (C)))
       (QUOTE (LAMBDA (X) X)))
```

vrijednosti parametra biti

L: **((A) (B) (C))**
FN: **(LAMBDA (X) X)**.

Kako nije zadovoljen uvjet **(NULL L)** izvršavala bi se druga grana cond izraza. Nakon ponovnog poziva funkcije **TESTR** vrijednosti parametara bi bile

```
L:  (A) .
FN: (FUNARG (LAMBDA()(CONS (CAR L) (FN)))
            ((L . (A) (B) (C)) (FN . (LAMBDA (X) X))))
```

LISP sistem tako raspolaže svim podacima potrebnima za izračunavanje.

Šta hoće McCarthy & Wooldridge?

Oni definiraju funkciju function (nema veze sa function u Lispu 1.5) oblika

$$function[[a_1; \ldots; a_n]; \langle expr \rangle; [s_1; \ldots; s_m]]$$

gdje su s_i varijable koje se koriste u izrazu <expr>, ili u funkcijama pozvanima kao rezultat izračunavanja <expr> a koje moraju biti izračunate u vrijeme u koje je funkcionalni argument izračunat.

$$function[[a_1; \ldots; a_n]; \langle expr \rangle; \mathbf{NIL}]$$

je interpretiran kao $\lambda[[a_1;\ldots;a_n]$ <expr>] i svi argumenti i parametri <expr> su evaluirani u trenutku <expr> je pozvan. To je ponašanje konzistentno sa ponašanjem čistog Lispa, prije dodavanja FUNCTION-FUNARG **„hacka.“**

Primjerice, funkcija TESTR bi bila definirana izrazom

```
(LAMBDA (L FN)
        (COND ((NULL L) NIL)
              (T        (TESTR (CAR L)
                               (FUNCTION ()
                                         (CONS (CDR L)
                                               (FN))
                                         (L FN))))))
```

.

Izračunavanje izraza

function[[a_1; …; a_n]; <expr>; [s_1; …; s_m]]

Generira i definira

(*Gsymbol* **(LAMBDA (**s_1 … s_m**)** <expr>**)**)

i vraća pointer na

(LAMBDA (a_1 … a_n**)(**Gsymbol **(QUOTE** s_1'**)** … **(QUOTE** s_m'**)))**,

gdje s_i' = *eval*[s_i]. Slijedeći pozivi se izvršavaju na isti način, samo što više nije potrebno generirati i definirati Gsymbol.

kraj