



THIS PAGE IS
INTENTIONALLY
LEFT BLANK.

Kazimir Majorinc

Hartovi makroi

Povijest Lispa 23.



Razmjena vještina
Hacklab u mami
2. ožujka 2013.

Makroi se javljaju prvo u assembleru. Kasnije COBOL.

<i>Location</i>	<i>Operation</i>	<i>Variable field</i>
M1	MAC	A, B, C, D, E
	OP1	A
B	E	C, R
	OP4	D
	END	
Any macro written		
	M1	S1, S2, S3, S4, S5
would generate		
	OP1	S1
S2	S5	S3, R
	OP4	S4

Greenwald, A Technique for Handling Macro instructions, **1959**.

Rana pojava makroa u Lispu, **McCarthyjev** metacircularni evaluator.

Primjerice

$$\begin{aligned} & eval[(F, A); ((F, (LAMBDA, (X), X)))] = \\ & = eval[((LAMBDA, (X), X), A); ((F, (LAMBDA, (X), X)))] \end{aligned}$$

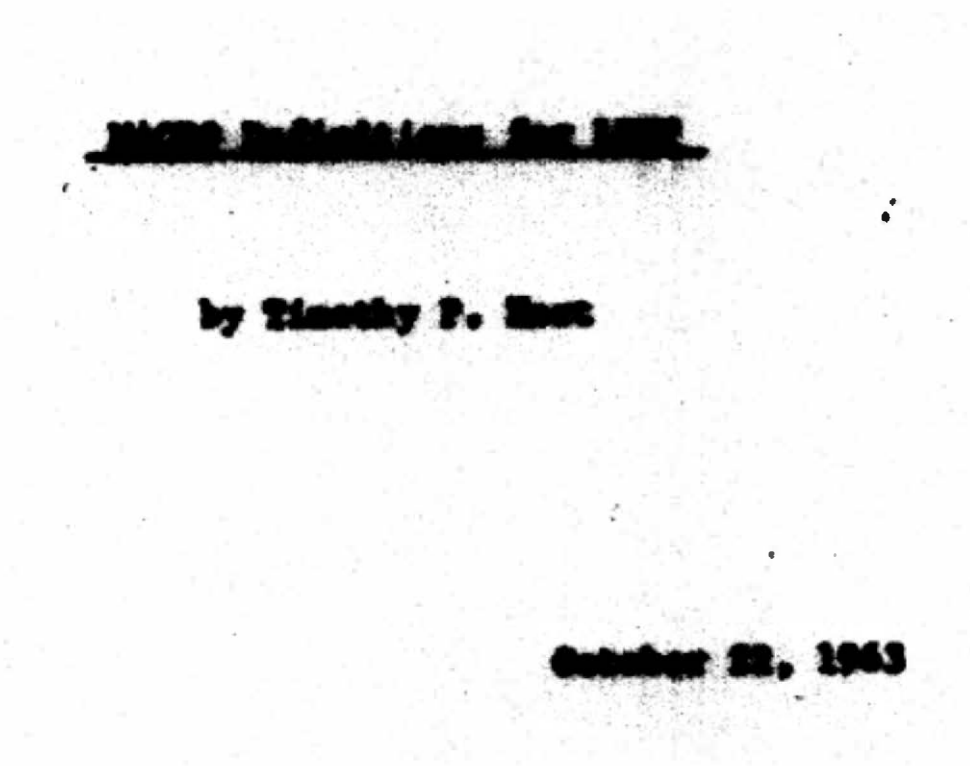
Dio u definiciji funkcije eval:

$$eval[e; a] = \dots \quad \mathbf{T} \rightarrow eval[\underset{a}{cons[assoc[car[e]; a]; cdr[e]}];$$

Specijalni, ali još uvijek makroi.

Timothy P. Hart, MIT-AIM 057, 22. listopad, 1963, *Macro Definitions in Lisp.*

Općeniti makroi.



Napisao nekoliko memoa i jedan konferencijski članak. Uspješan poduzetnik. U mirovini.

Po **Hartu**, makroi u LISPu mogu zadovoljiti dvije potrebe za koje su do tada korištene specijalne forme: definiranje operatora sa proizvoljnim brojem argumenata i sprečavanje evaluacije argumenata.

Primjerice, u Lispu 1.5 pseudo-funkcija *cset* kojom se definiraju konstante se često koristi u kombinaciji sa *quote*, u izrazima poput

(CSET (QUOTE X) 14).

Prvi programeri u Lispu su smatrali da bi bilo korisno izbjeći čestu primjenu specijalnog operatora *quote* uvođenjem novog operatora *csetq*, tako da se umjesto gornjeg izraza piše **(CSETQ X 14)**. Dakle, prvi parametar, **X**, ne smije se izračunavati. U Lispu 1.5 postojali su *fexprovi*, operatori koji su se pozivali sa ne-evaluiranim argumentima. Konkretno, postojao je baš i *fexpr csetq*.

Hart predlaže da se za takve svrhe uvedu makroi kao nova vrsta operatora, različita od funkcija i fexprova. Neka je m makro, a e_1, \dots, e_n neki simbolički izrazi. Poziv makroa

$$(m \ e_1 \ \dots \ e_n)$$

izračunavao be se u dvije faze. U prvoj, koju **Hart** naziva *širenje*, *ekspanzija* (engl. expansion) poziva se makro sa *neizračunatim* argumentima e_1, \dots, e_n . Makro m na temelju definicije m , “generira” novi simbolički izraz e . Primjerice, makro poziv **(CSETQ X 14)** bi se u prvoj fazi raširio u **(CSET (QUOTE X) 14)**. U drugoj fazi, dobiveni izraz e se izračunava.

Dvije faze ne moraju se izvršavati jedna za drugom. Po **Hartovom** opisu, pozivi makroa u definicijama funkcija širili bi se odmah čim su funkcije definirane. Za to je potrebno redefinirati funkciju `define` iz Lispa 1.5.

Nadalje, potrebno je odrediti način na koji korisnik može sam definirati makroe. **Hart** predlaže da se makroi definiraju kao funkcije sa jednim argumentom i to je, čini se, prvi puta da su makroi tako definirani u nekom programskom jeziku. Primjerice, makro *csetq* moglo bi se definirati kao

```
MACRO (((CSETQ (LAMBDA (FORM)
              (LIST (QUOTE CSET)
                    (LIST (QUOTE QUOTE)
                          (CADR FORM))
                    (CADDR FORM))))))1.
```

Prilikom poziva, parametru funkcije, u ovom primjeru **FORM**, se pridružuje, kao vrijednost cijeli makro poziv, u ovom primjeru **(CSET X 14)**.

¹ Ova se definicija makroa *setq* razlikuje od originalne **Hartove** u kojoj ima grešaka.

Drugi **Hartov** primjer je makro *stash*, koji dodaje vrijednost na početak liste. Makro poziv `(STASH e1 e2)` treba se *raširiti* u `(SETQ e2 (CONS e1 e2))`, gdje su e₁ i e₂ bilo koji izrazi. Jedna moguća definicija makroa *stash* je:

```
MACRO (((STASH (LAMBDA (FORM)
              (LIST (QUOTE SETQ)
                    (CADDR FORM)
                    (LIST (QUOTE CONS)
                          (CADR FORM)
                          (CADDR FORM)))))))).
```

Treći **Hartov** primjer, makro *enter*, koji kreira “točkasti par” i upisuje ga u *alistu*, ilustrira korištenje makroa pri definiranju novih makroa. Makro poziv

`(ENTER e1 e2 e3)`

treba se raširiti u `(STASH (CONS e1 e2) e3)`. Jedna moguća definicija makroa *enter* je dana meta-izrazom

```
MACRO
enter[form] = list[ STASH;
                    list[ CONS;
                          cadr[form];
                          caddr[form] ]];
                    caddr[form] ]
```

Četvrti **Hartov** primjer odnosi se na makro *select*, koji je već definiran i implementiran u LISPU 1.5 kao *fexpr*. Izraz

$$\text{select}[q; (q_1 e_1); (q_2 e_2); \dots; (q_n e_n); e]$$

se izračunava tako da se prvo izračuna q , a onda se redom izračunavaju q_1, \dots, q_n dok se ne pronađe q_i čija je vrijednost jednaka vrijednosti q . Tada se izračuna vrijednost odgovarajućeg e_i , i to je ujedno vrijednost cijelog izraza. Ako niti jedan od q_i nema vrijednost jednaku vrijednosti q , onda se izračuna vrijednost posljednjeg argumenta, e , i to je ujedno vrijednost cijelog izraza.

Hartov macro transformira gornji izraz u

```
((LAMBDA(g)(COND ((EQ g q1) e1)
                  ...
                  ((EQ g qn) en) (T e))))
q)
```

pri čemu je g simbol koji se ne pojavljuje niti u jednom od izraza $q, q_1, \dots, q_n, e_1, \dots, e_n, e$, čak niti indirektno.

Primjerice, ako izraz q je **(EVAL P)**, onda niti vrijednost **P** u trenutku izračunavanja ne smije biti S-izraz u kojem se pojavljuje g . To se postiže korištenjem LISP 1.5 funkcije *gensym* koja vraća potpuno novi, prethodno nekorišteni simbol.

Sam makro je presložen da bi ga ovdje iznijeli.

Uvođenje makroa zahtjeva promjene u Lisp sistemu. Prvo, treba definirati operator *macro*,

$$\text{macro}[l] = \text{deflist}[l; \text{MACRO}]$$

gdje je *deflist* funkcija iz Lisp 1.5 koja prima dva argumenta. Prvi je lista parova **((u_1 v_1)(u_2 v_2)...(u_n v_n))** gdje su u_i simboli a v_i lambda-izrazi. Drugi je „indikator.“ Na temelju toga, *deflist* kreira simbole i upisuje lambda-izraze i indikator u odgovarajuću „listu svojstava.“

Drugo, treba redefinirati funkciju *define*, tako da se odmah pri definiranju korišteni pozivi makroa rašire.

$$\text{define}[l] = \text{deflist}[m\text{def}[l]; \text{EXPR}]$$

Na kraju, zašto je **Hart** uveo makroe? Ne piše u memou, samo piše da može zamijeniti fexpr-ove, pogotovo što se fexpr-ovi ne mogu definirati u postojećem kompajleru, iako je ta mogućnost specificirana u Lisp 1.5 priručniku. Dakle, MIT zajednica je imala problema sa implementacijom fexpr-ova.