This page is intentionally left blank.

Kazimir Majorinc

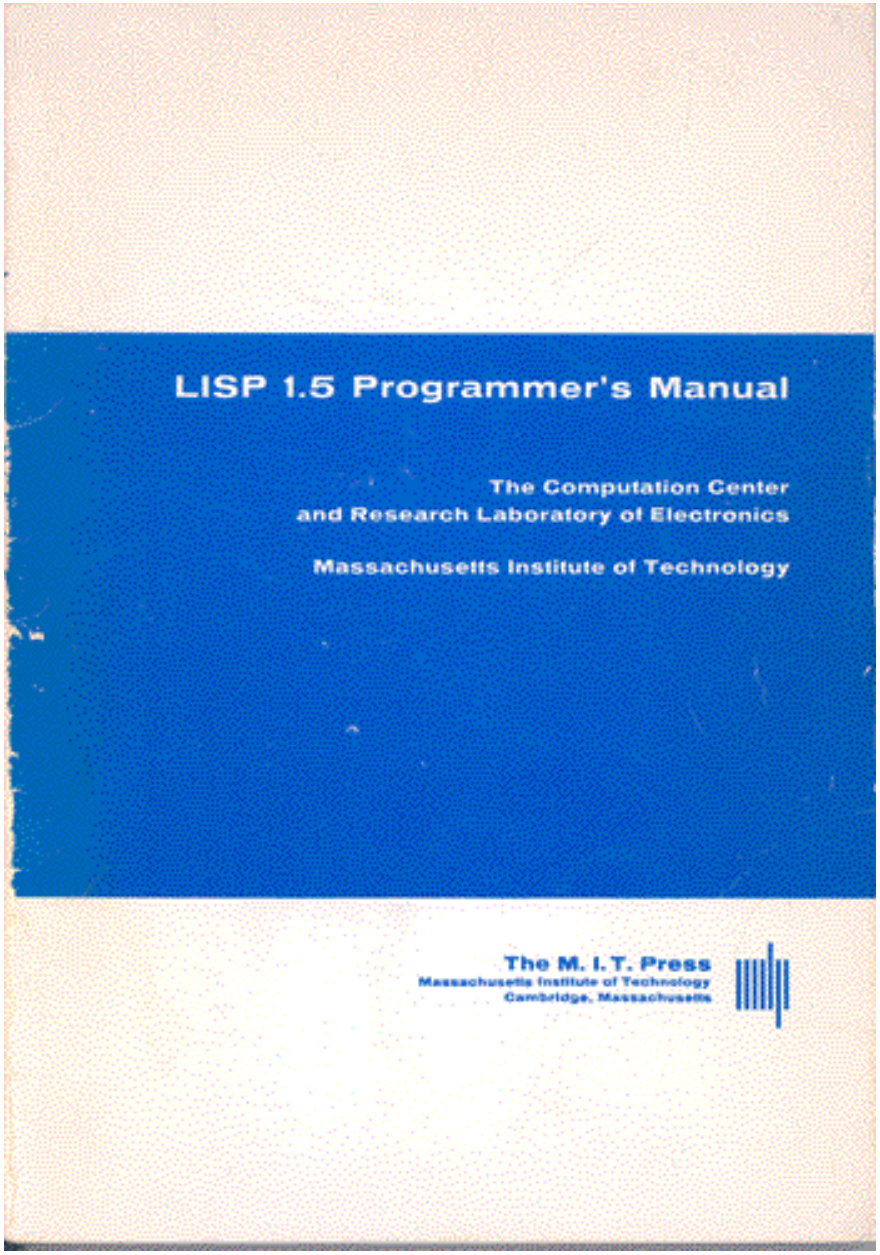
LISP 1.5 (I.)

Povijest Lispa 15.

t

Razmjena vještina
Hacklab u mami
8. prosinca 2012.

17. kolovoz, 1962.



John McCarthy - ukupni dizajn

Stephen B. Russell - intepreter

Paul W. Abrahams - manual, dijelovi kompajlera

Daniel J. Edwards - GC & aritmetika

Timothy P. Hart - kompajler

Michael I. Levin - kompajler, ovaj manual

Brayton i **Luckham** su već otišli.

327 citata.

LISP jezik.

Razlike između Lispa i ostalih programskih jezika:

(1) Svi podaci su S-izrazi; neodređene duljine, u obliku stabla tako da značajni podizrazi mogu lako biti izdvojeni.

(2) Jezik se sastoji od rekurzivnih funkcija nad simboličkim izrazima; notacija kojom se te funkcije zapisuju naziva se M-izrazi.

(3) Lisp, kao mašinski jezik, ali ne i većina viših jezika, može interpretirati i izvršavati programe u obliku S-izraza.

Rezime ideja objašnjenih u „Rekurzivne funkcije.“

U opisivanju jezika, McCarthy se ponekad koristi Backusovom notacijom. Primjerice,

<atomic-symbol>::=<LETTER><atom part>

<atom part>::=<empty>|<LETTER><atom part>

|<NUMBER><atom part>

Univerzalna LISP funkcija se zove evalquote

evalquote[*fn*;*args*] = *apply*[*fn*;*args*;NIL]

Okolina je sada oblika ((X.A) (Y.B)) umjesto ((X,A),(Y,B)).

```
apply[fn;x;a] =
      [atom[fn] → [eq[fn;CAR] → caar[x];
                   eq[fn;CDR] → cdar[x];
                   eq[fn;CONS] → cons[car[x];cadr[x]];
                   eq[fn;ATOM] → atom[car[x]];
                   eq[fn;EQ] → eq[car[x];cadr[x]];
                   T → apply[eval[fn;a];x;a]];
       eq[car[fn];LAMBDA] → eval[caddr[fn];pairlis[cadr[fn];x;a]];
       eq[car[fn];LABEL] → apply[caddr[fn];x;cons[cons[cadr[fn];
                                                  caddr[fn]];a]]]

eval[e;a] = [atom[e] → cdr[assoc[e;a]];
      atom[car[e]] →
                  [eq[car[e],QUOTE] → cadr[e];
                   eq[car[e];COND] → evcon[cdr[e];a];
                   T → apply[car[e];evlis[cdr[e];a];a]];
     T → apply[car[e];evlis[cdr[e];a];a]]
```

apply[CAR;(A);((A B))]??
apply je pomoćna funkcija

beskonačna evaluacija s-expr na mjestu operatora

pairlis obnavlja listu parova

evlis[(e1,...,en);a]= (eval[e1;a],...,eval[en;a])

U „the pure theory of LISP“ sve funkcije osim pet osnovnih trebaju se definirati svaki puta kada se koriste. To je neizvedivo u praksi. LISP ima puno više funkcija, kao i mogućnost da programer definira nove funkcije.

Elementarne funkcije, iako bi za neke slučajeve trebale biti definirane, u praksi uvijek imaju neku vrijednost.

Podržano je izračunavanje brojeva sa fiksnim i pomičnim zarezom. Brojevi se smatraju „pseudo-atomskim simbolima.“

II. LISP interpreter sistem.

Program:

```
DEFINE ((
(MEMBER (LAMBDA (A X) (COND ((NULL X) F)
        ( (EQ A (CAR X) ) T) (T (MEMBER A (CDR X))) )))
(UNION (LAMBDA (X Y) (COND ((NULL X) Y) ((MEMBER
        (CAR X) Y) (UNION (CDR X) Y)) (T (CONS (CAR X)
        (UNION (CDR X) Y))) )))
(INTERSECTION (LAMBDA (X Y) (COND ((NULL X) NIL)
        ( (MEMBER (CAR X) Y) (CONS (CAR X) (INTERSECTION
        (CDR X) Y))) (T (INTERSECTION (CDR X) Y)) )))
))
INTERSECTION ((A1 A2 A3) (A1 A3 A5))
UNION ((X Y  Z) (U V W X))
```

Funkcije se definiraju pseudo-funkcijom DEFINE. Pseudo-funkcija se izvršuje radi utjecaja na memoriju računala, ne radi rezultata. Vrijednost funkcije DEFINE u prošlom primjeru je (MEMBER UNION INTERSECTION).

Osnovna pravila za pisanje Lisp 1.5 programa:

(1). Programi su nizovi parova S-izraza; ti parovi su parovi argumenata za evalquote.

(2) Nema formata; svih 72 kolone bušene kartice mogu se koristiti

(3) Zarez je ekvivalentan razmaku. Mogu se pojavljivati bilo gdje, osim u sredini atomskog simbola.

(4) Ne koristi (QUOTE T), (QUOTE F) i (QUOTE NIL). Koristi T, F, NIL.

(5) Atomski simboli trebaju početi sa slovom

(6) Može se koristiti „dot notation.“ Razmaci se ignoriraju.

(7) Može se kombinirati „dot notation“ i liste. Npr.((A.B) C) itd.

(8) Oblik (A B C . D) je pokrata za (A (B (C . D))).

(9) Redoslijed kojim se funkcije definiraju je nebitan. Svaka funkcija može koristiti svaku drugu funkciju.