This page is intentionally left blank.

Kazimir Majorinc

Prva međunarodna Lisp konferencija (II.)

Povijest Lispa 22.

Razmjena vještina
Hacklab u mami
23. veljače 2013.

LEVIN, MICHEL.- Algebraic Compiler with LISP.

Mc. CARTHY, JOHN.- The LISP. 2 Compiler

Mc. INTOSH, HAROLD V.- The use of operator predicates in LISP.

MINSKY, MARVIN.- Garbage Collector methods

THOMAS, BILLY S.- Use of arrays in LISP. Group theory programs.

WE IZENBAUM, JOSEPH.- Open Ended compilation.

WILLIAMS, JOSEPH.- A LISP. Page plotter.

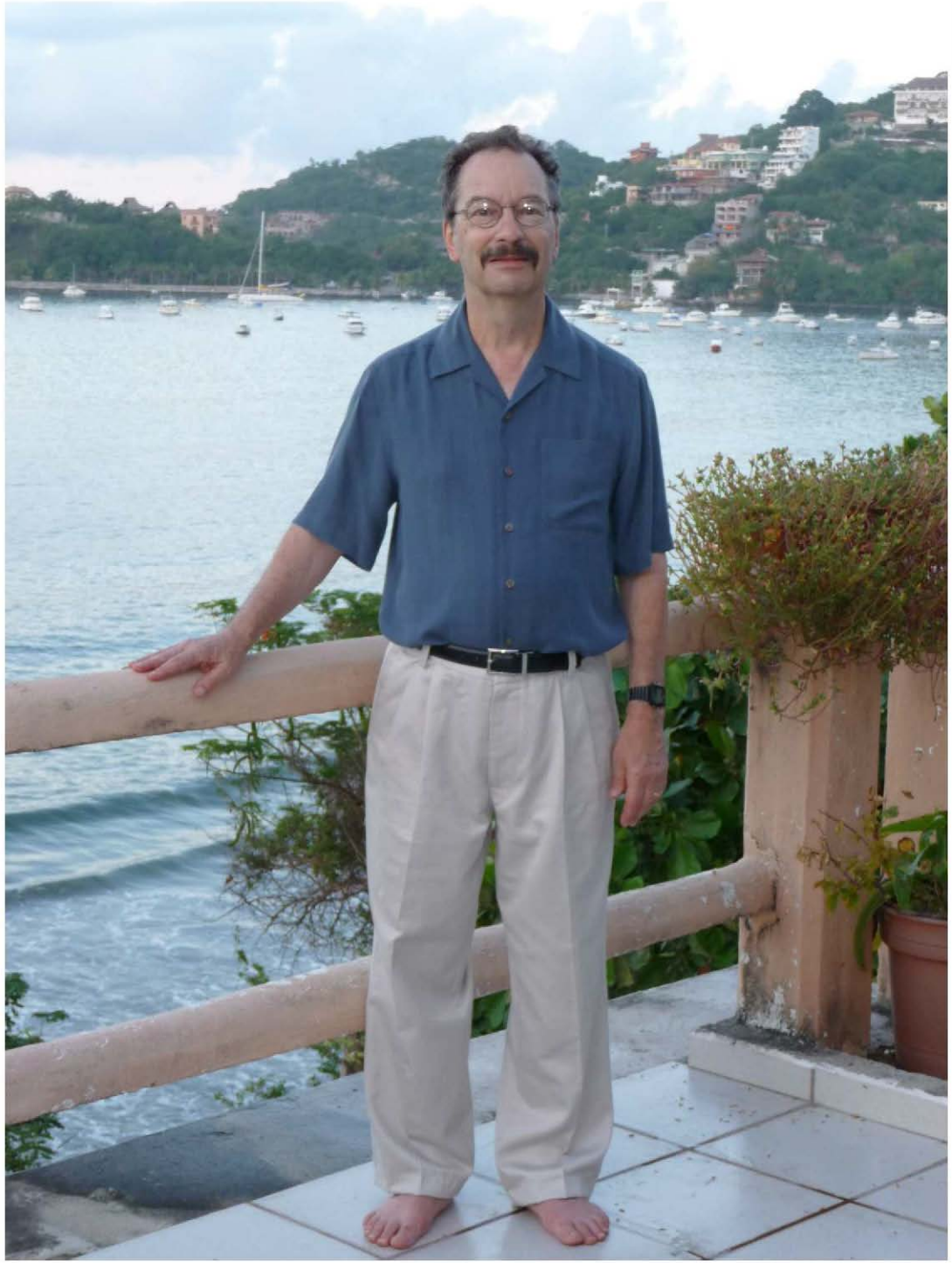
WOOLDRIDGE, DEAN.- An Algebraic Simplify Program in LISP.

YATES, ROBERT.- LISP. Group Analysis programs A LISP. Compiler for a variable word machine (Gamma 30 Scientific.)

RUSSELL - Debugging aids

VERHOVSKY - fro analyzers & similar December 15, 1963.

SFB'mg.



Michael Levin

autor drugog kompajlera za Lisp.

Paul McJones (Computer History Museum) pretpostavlja da se njegovo izlaganje odnosi na dokument MIT-AIM 039, Levin & Hart, The New Compiler.

Lisp sistem radi kao interpreter u okviru kojeg se može pozvati kompajler.

```
COMPILE ((FN1 FN2 ... ))
```

Kompajler, ako nije potreban može se izbrisati iz memorije.

Slobodne varijable (ne javljaju se samo unutar LAMBDA) moraju biti deklarirane

```
SPECIAL ((V1 V2 ...))
COMMON ((V1 V2 ...))
```

Funkcionalne konstante

```
(LAMBDA (X Y)
        (MAPLIST X
                 (FUNCTION (LAMBDA(J)
                                  (CONS (CAR J)
                                        X))))
```

Tri vrste varijabli: obične varijable se spremaju na stack; specijalne varijable imaju fiksno mjesto u memoriji; „common“ varijable se spremaju u alistu koja je dostupna svim funkcijama.

Kompajler u prvoj fazi procesira izvorni kod da bi ga učinio što efikasnijim. Rekurziju mijenja petljama, kad je moguće.

U drugoj fazi prevodi u asembler.

Treća faza postoji, ali nije opisana u memou. Sigurno prevodi asembler u mašinski kod.

Srpanj 1963:

STANFORD UNIVERSITY
Stanford, California

ARTIFICIAL INTELLIGENCE GROUP:

LISP 2 Specifications Conference

A G E N D A

(1) LISP 1.5 and its deficiencies
(a) Applications

(2) Proposals for LISP 2.0

(a) Linear Free Storage
(b) Numbers and other full words
(c) Auxiliary Storage
(d) Input language - infix notation
(e) Arrays
(f) Freer output format
(g) Sequence of implementation
(h) Comments
(i) Documentation and maintenance
(j) Hash Coding
(k) Subroutine linkage
(l) Storage conventions
(m) Effect of various I-O apparatus
(n) Interaction with programs in other languages
(o) Expressions having property lists
(p) Data structures
(q) Fitting into monitor

(3) Objections to LISP 1.5

1. Cluttered
2. Slow numberical calculations
3. Data structure
4. Slow interpreter

STANFORD ARTIFICIAL INTELLIGENCE PROJECT

Memo No.8
(Preliminary)

September 26, 1963.

STORAGE CONVENTIONS IN LISP 2

by John McCarthy

Storage conventions and a basic set of functions for LISP 2 are proposed. Since the memo was written, a way of supplementing the features of this system with the unique storage of list structure using a hash rule for computing the address in a separate free storage area for lists has been found.

Okupljeni na konferenciji pokušali su izbjeći stvaranje velikog broja novih tipova, ali nisu mogli postići suglasnost oko problema. Osim onoga što se danas smatra tipovima (cijeli broj, realni broj) htjeli su i informaciju može li se podatak premještati u memoriji - zbog garbage collectora.

Osnovni tip i dalje ostaje lista, ali riječi koje imaju negativni predznak (bit za predznak u IBM 904 riječi) označavaju tip podataka koji slijede. Time se ostavlja mogućnost razvoja sistema tipova. (Čini se da su tipovi vezani za liste.)

Osim car i cdr, uvodi se opet funkcija cwr - koja vraća sadržaj cijele riječi.

Moguće je izračunavati adrese podataka, na primjer

caddr[x + 5] := (PLUS A B)

Gornja naredba je nesigurna.

Uvode se funkcije

mk1[n; type; w1; ....; wn]

mk2[n; type; a1; d1; ...; an; dn]

mk3[n; type; r1, ... rn; w1; ...; wn]

mk3[n; type; r1; ... rn; a1; d1; ...; an; dn]

koje spremaju podatke i vraćaju adresu na kojoj su spremljeni,

declare1[n; type]

declare2[n; type]

declare3[n; type; r1;...; rn]

koje samo rezerviraju mjesto u memoriji i vraćaju adresu.