This page is intentionally left blank.

Kazimir Majorinc

Algebarski jezik za obradu simboličkih izraza (I)

Povijest Lispa 3.

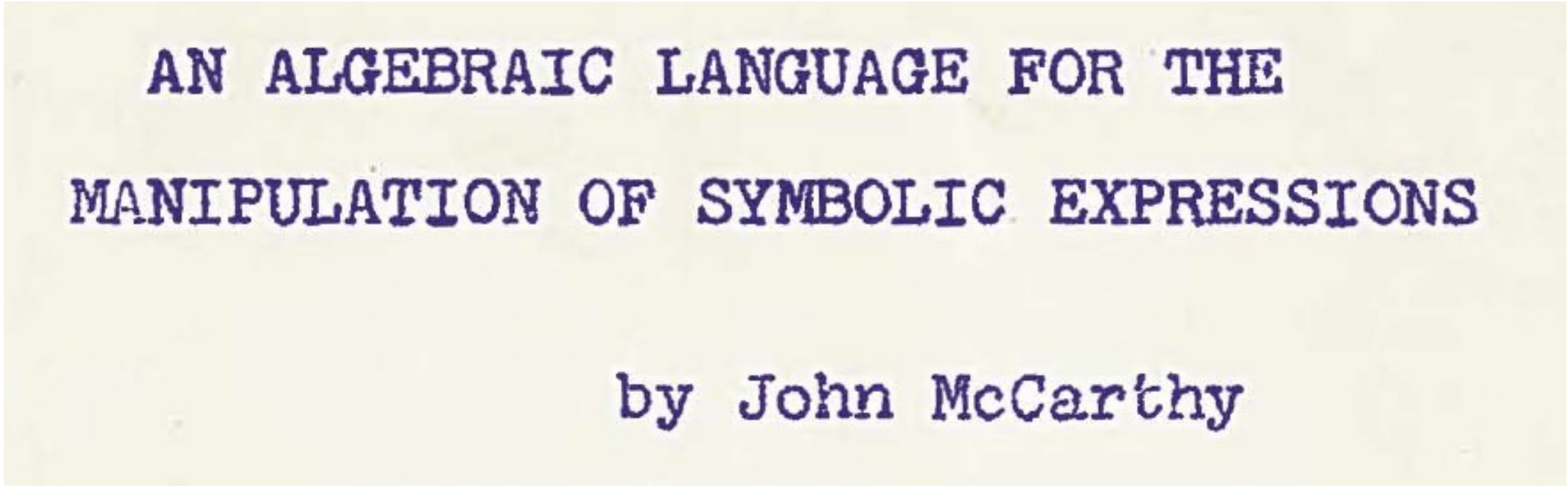
Razmjena vještina
Hacklab mama
21. srpnja 2012.

Do sada:

Prijedlog Ljetnog projekta iz 1957. (filozofski)

Prijedlog kompajlera, prosinac 1957. (dinamičan, refleksivan, skoro pa paralelan, ali još nije Lisp)

Rad na Algolu 58 (uvjetno grananje, makro naredbe.)


AN ALGEBRAIC LANGUAGE FOR THE

MANIPULATION OF SYMBOLIC EXPRESSIONS

by John McCarthy

Šapirografirani dokument, 20 strana, rujan 1958. Službena oznaka AIM-001. Izgubljen na MIT. Pronađen u privatnoj zbirci i objavljen na “Software preservation museum” 2010.

To je već LISP, samo još ne precizno definiran i lijepo izložen.

Primjene jezika.

Jezik je pogodan za program *Advice taker*, dokazivanje teorema, pojednostavljenje formula, simboličko deriviranje i integriranje, pisanje kompajlera, heurističkih programa, predstavljanje izraza čiji broj i duljina variraju na teško predvidive načine.

Jezik nije pogodan za predstavljanje lista fiksne duljine gdje se često želi direktno pristupiti n-tom elementu.

Svojstva jezika

Implementacija izraza kao lista. Izrazi su predstavljeni kao liste čiji svaki element zauzima jednu riječ u računalu. Svaki element liste sastoji se od podatka i adrese slijedećeg elementa liste. Za zadnji element liste, adresa slijedećeg elementa liste je 0. Posebno, podatak može biti i adresa prvog elementa neke liste.

Riječ IBM 704 računala ima 36 bita. Podijeljena na dva dijela od po 15 bita (adresa i dekrement) i dva dijela od po 3 bita (prefiks i oznaka (tag)). Podatak je spremljen u adresi, adresa slijedećeg elementa u dekrementu.

Algebarska notacija. Izlaz jedne “procedure” je ulaz druge, pa se može izbjeći davanje imena međurezultatima. Redoslijed kojim je “funkcionalna formula” zapisana dozvoljava nam razmišljanje od željenog rezultata ka komponentama.

Rekurzija. Kad podprogram pozove sam sebe valja zaštititi međurezultate od promjena. To se može učiniti spremanjem međurezultata u liste.

Uvjetni izrazi. U dosadašnjim programskim jezicima nisu odgovarajuće riješeni. [Naredba **if** se danas ne čini puno lošija od uvjetnih izraza, no tada se morala kombinirati sa **go to**].

Vrste kvantiteta [“quantities”]. Liste nisu kvantiteti. Iako se na listama mogu definirati mnoge korisne operacije, većina računanja se mora definirati nad pojedinačnim elementima. “*It still seems to be necessary to compute with the addresses of the elements of the lists*”.

Cjelobrojni kvantiteti [integer.] Bez tipografskih konvencija za razlikovanje.

Cijele riječi. Autor nije siguran trebaju li mu brojevi sa pomičnim zarezom.

Iskazni kvantiteti. Vrijednost može biti 0 i 1. Definirat će se operacije $\wedge$, $\vee$ i **~**. Predikati su funkcije koje imaju vrijednosti 0 i 1.

Lokacijski kvantiteti. Točka [point] u programu može biti označena i adresa te točke se naziva lokacijski kvantitet. Izračunavanje sa tim kvantitetom je ograničeno.

Funkcijski kvantiteti. "Ovi će sigurno biti dozvoljeni kao parametri potprograma, ali njihove pune mogućnosti neće se moći iskoristiti u ranom sistemu."

Vrste naredbi

Najvažnija, **aritmetička naredba** ili **naredba za zamjenu** ima oblik *a*`=`*b*, gdje

a je varijabla, ili

a je polje sa indeksom, npr `A(`*i*`)`, ili

a je `cwr(`*i*`)`, `car(`*i*`)`, `cdr(`*i*`)` i mnoge druge

gdje je *i* izraz koji se izračunava u cjelobrojni kvantitet, a *b* je bilo koji izraz.

Prijenos kontrole. Naredba `go(`*e*`)` prenosi kontrolu na lokaciju dobivenu izračunavanjem lokacijskog izraza e.

Set naredba. `set(A;` *q*1`,` ...`,` *qm*`)` definira polje (ne listu!) `A` sa vrijednostima dobivenim izračunavanjem *q*1, ..., *qm*, koje, posebno, mogu biti i lokacijski kvantiteti.

Pozivanje podprograma. `s(`*expr*1`,` ...`,` *exprn*`)`. Ne treba `CALL` kao u Fortranu.

Deklarativne rečenice (“sentences.”) Dva oblika.

`(a; p1, ..., pn)` unosi izraze *p*1, ..., *pn* u “*property list*” pridruženu simbolu *a*. Svaki simbol u programu ima takvu listu koju koristi kompajler, a može se koristiti i direktno u programu.

`(a1, ..., an, p)` unosi izraz *p* u “property list” svih simbola *a*1, ..., *an*.

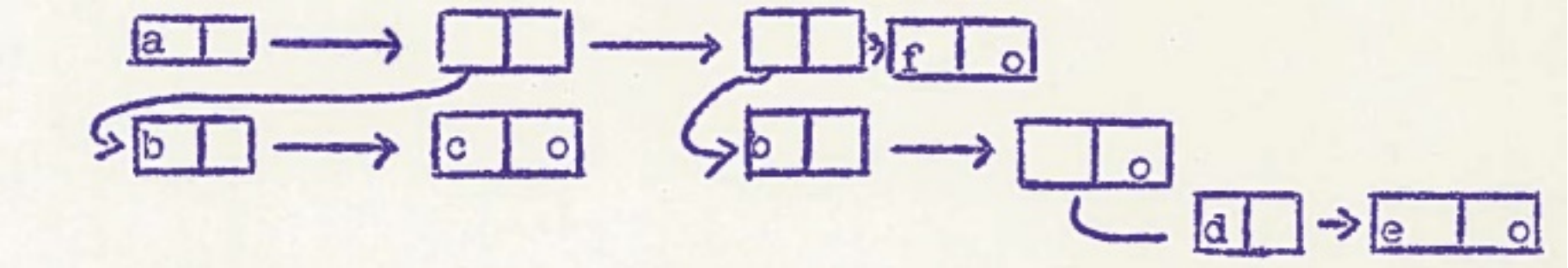
Složene naredbe. Niz naredbi može biti zatvoren u “zagrade” i dobiti ime. Sintaksa još nije određena.

Naredba za iteraciju. McCarthy nije još odlučio o obliku te naredbe, ali traži nešto poput do u Fortranu (cjelobrojni indeksi, **for** u većini jezika) ali i iteriranje kroz listu.

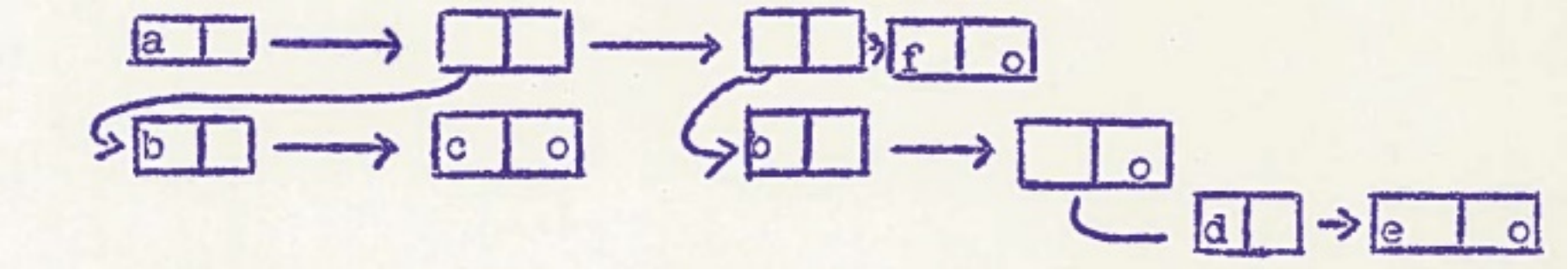
Definicija podprograma (subroutine). Parametri uključuju funkcije, potprograme i lokacijske izraze. Potprogrami mogu biti kompajlirani ili u programu, ili nezavisno.

Liste.

Predstavljanje izraza. Primjerice, niz (“sequence”) **`(a, (b,c), (b, (d,e)),f)`** je predstavljen u memoriji na slijedeći način:



Problem: je li podlista **(b, c)** dio liste? Ako želimo brisati listu, želimo li obrisati i podlistu?



Oba odgovora mogu imati smisla. Zato se u svakoj riječi izdvajaju prva dva bita i slijedi konvencija:

00 Ako je podatak podlista, ne briše se sa listom.

01 Podatak nije podlista i briše se sa listom.

10 Podatak jest podlista i briše se sa listom.

Funkcionalni izraz `f(e1,...,en)` je predstavljen nizom (sequence) `(f,e1,...,en)` i odgovarajućom strukturom u memoriji računala. Izraz `a+b+c` je predstaljen nizom `(+,a,b,c)`.

Funkcije su zapravo "funkcije", jer vrijednost izračunavanja ovisi o stanju memorije i mogu mijenjati vrijednost argumenata.

Sistem održava **listu neiskorištenih riječi** (free storage list). Potprogrami uzimaju adrese riječi sa liste i vraćaju je na nju.

Riječ računala IBM 704. Riječ ima 36 bitova koji se grupiraju i označavaju na slijedeći način:

w - cijela riječ

p - prefiks, bitovi 0-2

s - sign, bit 0

i - indikator, bitovi 1-2

d - dekrement, bitovi 3-17

t - tag, bitovi 18-20

a - adresa 21-35

Osnovne funkcije za izdvajanje dijelova riječi.

`bit(`*w*`,`*n*`)`** vraća *n*-ti bit riječi *w*

`seg(`*w*`,`*m*`,`*n*`)`** vraća segment bitova od *m* do *n*.

Izvedene funkcije za izdvajanje dijelova riječi.

dec(*w*)=**seg**(*w*,**3**,**17**)

add(*w*)=**seg**(*w*,**21**,**35**)

slično **ind**, **sgn**, **tag** ...

Funkcije koje indirektno izdvajaju dijelove riječi.

cwr(*n*) je riječ na adresi *n*

cdr(*n*) je ekvivalentna **dec**(**cwr**(*n*))

car(*n*) je ekvivalentna **add**(**cwr**(*n*))

slično **cir**, **csr**, **ctr** ...

Funkcije **cwr**, **cdr**, **car** i ostale se mogu koristiti na lijevoj strani naredbe za zamjenu

cwr(55)=1000.

Funkcije koje spajaju dijelove riječi.

comb 4(*p***,** *d***,** *t***,** *a***)**

comb 5(*s***,** *i***,** *d***,** *t***,** *a***)**

Funkcije za konstrukciju.

Izraz **`consw(`***w*`)` sprema vrijednost w u prvu slobodnu riječ, vraća adresu te riječi, i briše adresu sa *liste neiskorištenih riječi.*

Funkcija za brisanje riječi.

Izraz **`erase(`***n*`)` samo vraća adresu *n* na listu neiskorištenih riječi.

Idući put: nešto naprednije funkcije i čak nekoliko programa.