This page is intentionally left blank.

Kazimir Majorinc

ADVICE TAKER.

Povijest Lispa 5.

Razmjena vještina
Hacklab u mami
15. rujna 2012.

Početkom rujna **1958**. **McCarthy** počinje razvoj programskog jezika koji će postati Lisp. U prvom “memou” najavljuje da će jezik biti pogodan za “manipuliranje rečenicama” neophodno za dokazivanje teorema i “*advice taker project*” (malim slovima!)

O *Advice takeru* se malo pisalo. Najvažniji je dokument članak u zborniku skupa “*Mechanisation of thought processes*”, 24-27. studenog **1958**. na kojem je **McCarthy** predavao na temu “*Programs with common sense*”. Osam stranica + šest strana diskusije. Kasnije se rijetko spominje, uz isticanje velikog povijesnog značaja.

“Advice taker” je predloženi program za rješavanje problema manipuliranjem rečenica u formalnim jezicima.

Program se razlikuje od prethodnih programa time što ne bi izvodio samo konkluzije, nego i “heuristiku” (praktične metode za traženje rješenja problema); termin uveo **Polya** (1945), preuzeo **Newell**.)

Glavna prednost “*Advice takera*” je da se ponašanje programa može poboljšati bez izmjena programa. Korisnik treba upisivati sudove u formalnom jeziku, ali ne mora mijenjati pa niti razumjeti program.

Program ima “zdrav razum” ako automatski izvodi dovoljno široku klasu direktnih logičkih posljedica svega što mu je rečeno i što već zna. “*Advice taker*” bi imao to svojstvo.

“*Advice taker*” bi mogao izvoditi sva moguća pravila ponašanja (heuristike?) i isprobavati ih. Teškoće sa takvim pristupom su: zanimljiva pravila ponašanja su rijetka, i male, korisne promjene u ponašanju se ne mogu izraziti malim promjenama u pravilima ponašanja.

Svojstva sustava koji evoluira do ljudske inteligencije

Svi oblici ponašanja se mogu predstaviti u sustavu. Dakle, sustav treba imati mogućnost pisanja programa u nekom općem programskom jeziku.

2. Zanimljive promjene u ponašanju moraju biti izrazive na jednostavan način.

3. Svi aspekti ponašanja osim potpuno rutinskih trebaju se moći poboljšati. Posebno, mehanizam za poboljšanje mora se moći poboljšati.

4. Stroj mora imati ili razviti ideju polu-uspjeha, jer je to najčešći ishod eksperimenta.

5. Sistem mora biti sposoban kreirati potprograme koji se mogu uključiti u druge programa. “Učenje potprograma” je složeno jer potprogrami nisu jednostavno dobri ili loši, nego su takvi pod određenim uvjetima.

U vrijeme konferencije, projekt se koncentrirao na drugo svojstvo: zanimljive promjene u ponašanju moraju biti izrazive na jednostavan način - *onako kao što to mogu raditi ljudi*.

Deklarativne i imperativne rečenice

Glavna razlika između modificiranja ponašanja računala i čovjeka je u vrsti rečenica. Ponašanje računala se modificira pretežito imperativnim, a ponašanje ljudi deklarativnim rečenicama. **McCarthy** ne definira te dvije vrste rečenica)

Prednosti imperativnih rečenica:

1. Brže se izvršavaju
2. Nemaju pretpostavku prethodnog znanja

Prednosti deklarativnih rečenica:

1. Koriste prethodno znanje
2. Iz deklarativnih rečenica se može izvoditi novo znanje
3. Manja ovisnost o redoslijedu unošenja
4. Manja ovisnost o prethodnom stanju računala

Glavna svojstva Advice takera

Metod za reprezentaciju izraza u računalu.

1. Svaki term je izraz. (Termi nisu definirani)
2. Niz izraza je izraz.

2. Samo jedno pravilo izvođenja, kombinacija supstitucije i modus ponensa **(*F* -> *G*) i *F* => *G*.**

3. Potprogram za “neposrednu dedukciju”. Za zadan skup premisa izvodi neposredne konkluzije. Isprva samo konkluzije koje se mogu u jednom koraku izvesti iz premisa, kasnije će se ponašanje potprograma unaprijediti. Konkluzije se izvode samo logičkim putem, dakle bez obzira na semantiku. U tom potprogramu se nalazi inteligencija Advice takera.

4. Izrazi mogu biti deklarativne rečenice, ali i imena entiteta i objekata. Entiteti su objekti o kojima se nešto može reći. Za većinu ljudi broj 3812 nije entitet, jer ne mogu reći ništa o tom broju, osim onoga što slijedi iz strukture broja. Broj 1776 je entitet za one koji znaju da je te godine bila Američka revolucija.

5. U sistemu su osim deklarativnih rečenica predstavljene individue, funkcije i programi.

6. Program se izvršava cilički. Potprogram za neposrednu dedukciju je primjenjen na listu premisa i listu individua. One konkluzije koje imaju oblik imperativnih rečenica se izvršavaju. Posebno, i taj ciklus može biti imperativna rečenica.

Primjer

ssume that I am seated at my desk at home and I wish to go to the airport. My car is at my home also. The solution of the problem is to walk to the car and drive the car to the airport.

1. Popišemo premise

at(I,desk)

at(desk,home)

at(car,home)

at(home,county)

at(airport,county)

at(x,y),at(y,z) -> at(x,z) ili

transitive(at)

transitive(u) -> (u(x,y),u(y,z) -> u(x,z))

2. Pravila za hodanje i vožnju:

walkable(x),at(y,x),at(z,x),at(l,y) -> can(go(y,z,walking))

drivable(x),at(y,x),at(z,x),at(car,y),at(l,car) ->

can(go(y,z,driving))

walkable(home)

drivable(county)

3. Svojstvo odlaska

did(go(x,y,z)) -> at(l,y)

4. Problem zadan premisom

want(at,I,airport)

5. Predikat canachult (ako se u situaciji x može primjeniti i primjeni y onda dolazimo u situaciju u kojoj se može primjeniti z)

(x->can(y)),(did(y)->z)->canachult(x,y,z)

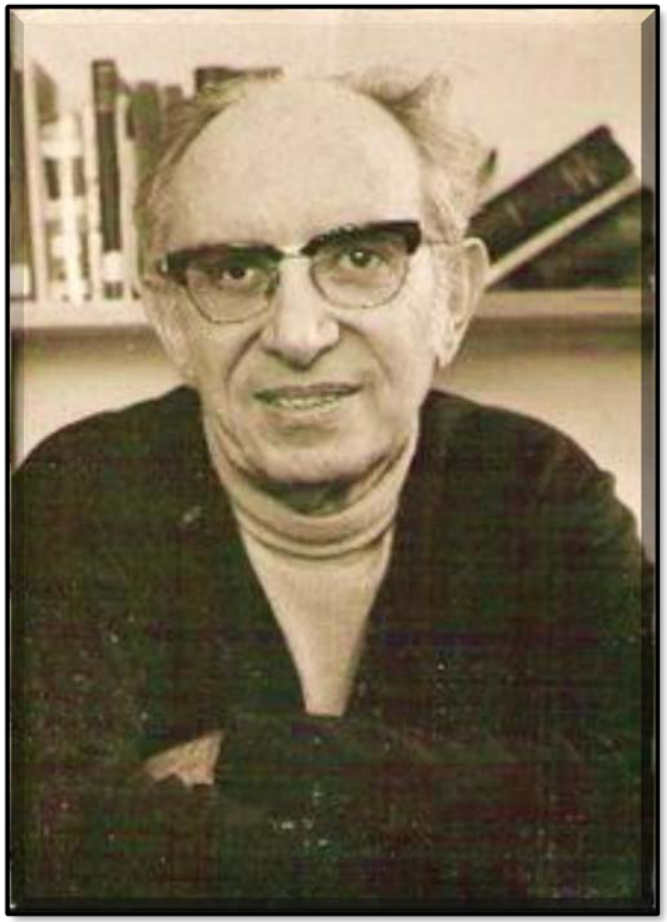
canachult(x,y,z),canachult(z,u,v)->canachult(x,prog(y,u),v)

x,canachult(x,prog(y,z),w),want(w) -> do(y)

Iz prethodno navedenog valja izvesti

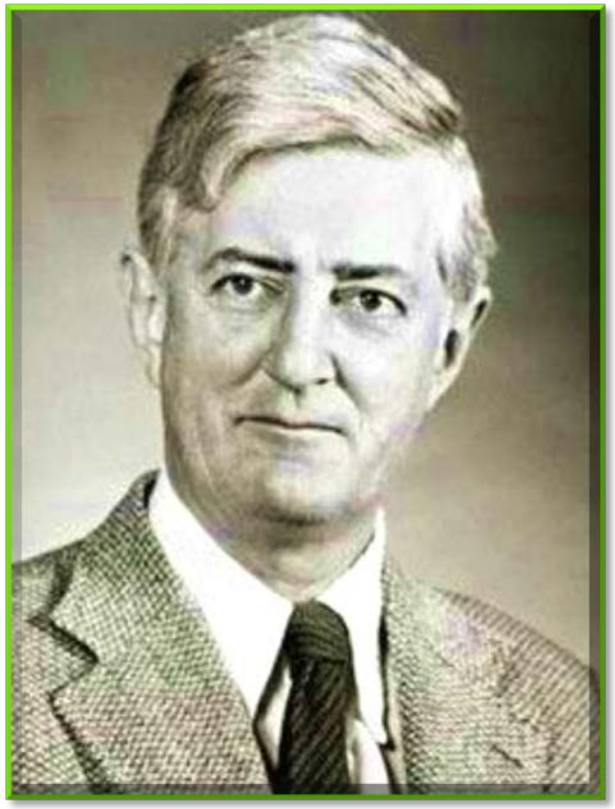
do(go(desk,car,walking))

Diskusija



Yehoshua Bar-Hillel (1915-75) oštro kritizira **McCarthyja**. Nejasne filozofijske i pseudofilozofijske pretpostavke. **McCarthy** misli da ako opiše svoj program sa dovoljnim nemarom, da će onda program moći rješavati različite probleme. Primjer problema je prejednostavan (npr. **at** ne može biti tranzitivan, vrijeme nije uzeto u obzir), a čak niti za takav primjer nije objašnjeno kako bi do zaključivanja zaista došlo. Problem koji bi uvjerio **Bar Hillela** bio bi nekoliko redova veličine složeniji.

Oliver G. Selfridge (1926-08), bivši profesor Minskog, kritizira sa suprotnog polazišta. Ljudi gotovo ništa ne zaključuju deduktivnom logikom; ona se koristi samo za posebno “svete” svrhe. **McCarthy** priznaje da program sada ne radi, no pitanje je samo može li ga popraviti.



Bar Hillel: Možda, ali **McCarthy** želi biti logički korektan.

McCarthy: Kritika prejaka, mi tek radimo prve korake.

Što se dogodilo s Advice takerom?

ema podataka. Program je spomenut u nevelikom broju publikacija - našao sam samo dvije-tri, poput **Russell** & **Norvig**, 2010, gdje mu se pripisuje veliki značaj, ali se ne spominje što je sa programom bilo.

McCarthy je 2003. održao predavanje "*The advice taker revisited*", slajdovi na njegovom sajtu u kojem piše na isti način - piše isključivo o članku, ali ne spominje program, ne izvodi nikakve zaključke iz rada ili pisanja stvarnog programa - što je prilično siguran znak da program nikad nije napisan.

Zaključak

McCarthy iznosi ideje, a ne rezultate. Ideje su krajnje ambiciozne. McCarthy je filozofijski i praktično orijentiran; ne spominje nikakvu matematičku teoriju (svojstva predikatnog računa itd.) Iznešene ideje dovoljno dobro motiviraju dizajn jezika kao što je Lisp.