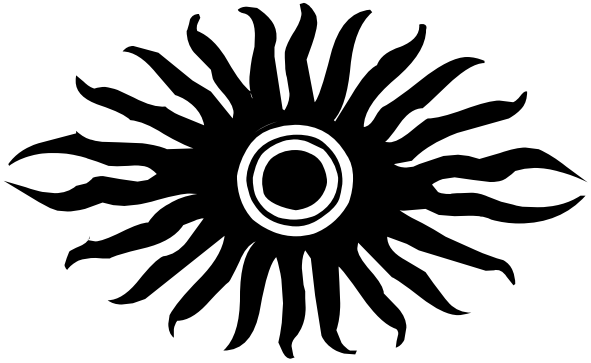
This page is intentionally left blank.

Kazimir Majorinc

S-FUNKCIJE I TURINGOVI STROJEVI

Povijest Lispa 12.



Razmjena vještina

Hacklab u mami

10. studeni 2012.

MIT Research Laboratory of Electronics, Quarterly Progress Report, 15. travnja, 1959.

Sadrži jednu od bar četiri „grube verzije“ članka „*Recursive functions of symbolic expressions and their computation by machine. Part I.*“ Za razliku od ostalih verzija, ova, treća, sadrži i poglavlje „*Computability and unsolvability.*“ Hipoteza: to je trebao biti *part II*. koji nikad nije napisan.

Svrha poglavlja je pokazati da su S-funkcije pogodnije za razvoj „teorije rekurzivnih funkcija“ nego li Turingovi strojevi i rekurzivne funkcije na cijelim brojevima, jer:

(1). U teoriji, funkcije treba kodirati u elemente na kojima je funkcija definirana. S-funkcije se vrlo prirodno kodiraju u S-izraze.

(2) Zanimljive S-funkcije su izračunljive, razumno brzo, na suvremenim računalima.

(3) Korištenje uvjetnih izraza pojednostavljuje rekurzivne definicije.

Glavni rezulati teorije izračunljivosti su (u terminima turingovih strojeva):

1) Turingova teza: svaki „efektivno izračunljiva funkcija“ može biti predstavljena Turingovim strojem.

2) Postoji „univerzalni Turingov stroj“ koji može simulirati bilo koji drugi stroj.

3) Ne postoji Turingov stroj koji rješava problem „da li se Turingovi strojevi zaustavljaju.“

Treba ih pokušati izreći u terminima

S-funkcija koja simulira Turingove strojeve.

Može li se zaista svaka „efektivno izračunljiva funkcija“ predstaviti S-funkcijom? Ne znamo; jer ne znamo što su to „efektivno izračunljiva funkcija“.

Ali, može se dokazati da su S-funkcije „bar toliko dobre“ kao Turingovi strojevi. Za svaki Turingov stroj se može definirati funkcija koja izračunava isto što i Turingov stroj.

1. Pokazati kako se konfiguracija T-stroja prevodi u S-izraz
2. Definirati S-funkciju *succ* koja zadanoj konfiguraciji pridružuje slijedeću konfiguraciju
3. Na temelju tako definiranog *succ* definirati S-funkciju *turing* koja izračunava isto što i turingov stroj.

Turingov stroj služi za čitanje i pisanje na obostrano beskonačnoj, uglavnom praznoj traci. Čita i piše konačno različitih simbola; ima jedno od konačnog broja mogućih unutrašnjih stanja.

Rad svakog Turingovog stroja opisan je popisom petorki koje se sastoje od

1. stanje stroja
2. simbol koji se učitava
3. simbol koji se ispisuje
4. pravac kretanja (L ili R)
5. novo stanje

Svakoj petorci se može pridružiti S-izraz. Primjerice **`(0,1,B,R,0)`**.

Stroj se zaustavlja ako naiđe na kombinaciju stanja stroja i učitanog simbola koji nisu „na popisu“.

Turingov stroj predstavljen je u potpunosti S-izrazom

(*početno stanje***,***prazni simbol***,***petorka*$_1$**,...,***petorka*$_k$**)**

Primjerice,

```
(0,B,(0,0,B,R,0),(0,1,B,R,1),(0,B,0,R,2),
     (1,0,B,R,1),(1,1,B,R,0),(1,B,1,R,2))
```

Ovaj se Turingov stroj zaustavlja kad „uđe u stanje 2“.

„Konfiguraciju“ Turingovog stroja čine promjenjive vrijednosti, tj. unutrašnja vrijednost stroja, traka i položaj glave za čitanje.

Primjerice, traka i položaj glave za čitanje ...1101[0]110 ... se mogu predstaviti S-izrazom

```
(0,(1,0,1,1),(1,1,0))
```
.

Promjenjivi dijelovi, tj. unutrašnje stanje stroja i traka se mogu predstaviti četvorkom, tzv. **konfiguracijom**. Primjerice

```
(0,0,(1,0,1,1),(1,1,0))
```
.

McCarthy definira funkciju *succ*[*stroj*;*konfiguracija*].

Primjerice,

succ[**(0,B,(0,0,B,R,0),(0,1,B,R,1),(0,B,0,R,2),**
(1,0,B,R,1),(1,1,B,R,0),(1,B,1,R,2))];
(0,0,(1,0,1,1),(1,1,0))] =

= **(0,1,(1,0,1,1,B),(1,0))**.

Program je kompliciran, ali rutinski.

U slijedećem koraku definira funkciju *turing*[*stroj*,*traka*] koja od stroja i trake konstruira **konfiguraciju** i rekurzivno poziva *succ*.

Pitanje o S-funkcijama neodlučivo S-funkcijom.

S-funkcije su definirane, ovisno o tome da li izračunavanje rekurzije završava.

Bilo bi korisno definirati S-funkciju *def*[*f*;*s*] , koja izračunava **T** ako izračunavanje *f*O[*s*] završava, **F** inače, ZA SVAKI fO i s.

Teorem: def[f;s] nije S-funkcija. **Dokaz.** Pretpostavimo da jest. Definirajmo funkciju g.

$$g=\lambda[[f];[\sim def[f;f]\rightarrow \mathbf{T}, \mathbf{T}\rightarrow car[\mathbf{NIL}]]].$$

Koliki je g[g*]?

Ako je g[g*] definiran, onda po definiciji g, ~def[g*;g*]=**T**.

Ali, po definiciji def, to znači to znači da g[g*] nije definirana.

Obratno, ako g[g*] nije definirana, onda po definiciji def, def[g*;g*]=**T**. Tada g[g*] jest definirana.