SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Matematička logika i izračunljivost

Halting problem

Student: Petra Franić

JMBAG: 0036420512

Zagreb, studeni, 2009.

Sadržaj

1. Uvod

Izračunavanje je proces kod kojeg iz nekih početno danih objekata s fiksiranim skupom pravila dobivamo krajnji rezultat. Početne objekte nazivamo ulazni podaci, fiksirani skup pravila naziva se algoritam ili program, a krajnji rezultati se nazivaju izlazni podaci.

Ne zahtijeva se da za sve ulazne podatke svaki algoritam daje izlazni rezultat. To znači da neki algoritam za neke ulazne podatke može računati, a da nikad ne stane.

Osvrtom na Church-Turingovu tezu koja iskazuje da je za svaki algoritam moguće izgraditi Turingov stroj koji ga izvršava, u nastavku će Turingov stroj biti detaljnije obrađen i osnova za dokaz definiranih problema.

Ispostavlja se da je Turingov stroj izuzetno moćan model automata te svi pokušaji izmjene definicije Turingovog stroja u svrhu stvaranja moćnijeg stroja nisu rezultirali uspjehom.

Turingov stroj omogućava izvođenje algoritama, a dodatno olakšava i analizu složenosti zbog svoje jednostavne izvedbe. Primjenom Turingovog stroja kao idealnog modela definirani su mnogi moderni problemi vezani uz analizu algoritama, kao npr. Halting problem čija nerješivost će biti dokazana u nastavku.

Halting problem se bavi spomenutom činjenicom da algoritam za neke ulaze ne mora dati rezultat te pokrenemo li ga s nekim parametrima ne postoji način da saznamo hoće li se izvoditi u beskonačnost ili mu samo treba još vremena.

2. Turingov stroj

Turingov stroj je imaginarni uređaj koji može simulirati logiku bilo kojeg računalnog algoritma i od velike je pomoći u ispitivanju granica izračunljivosti. Opisao ga je Alan Turing koji se uvelike interesirao za pojam izračunljivosti. Intuitivno gledajući zadatak je izračunljiv ako je moguće specificirati niz instrukcija čijim praćenjem se dolazi do izvršenja zadatka.

2.1. Formalna definicija Turingovog stroja

Turingov stroj (TS) formalno se zadaje uređenom sedmorko:

$$M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$$

gdje je:

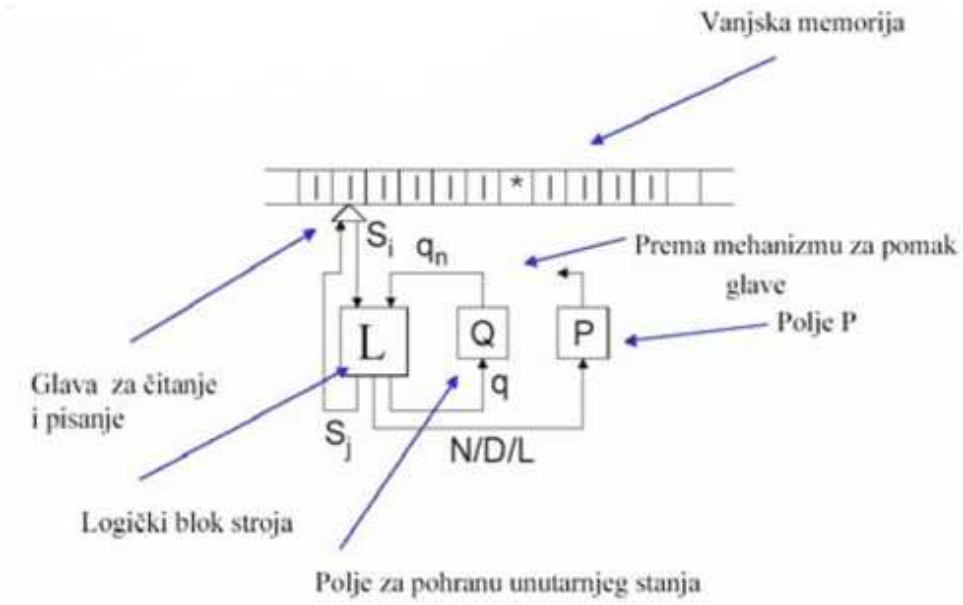
- Q konačan skup stanja
- Γ konačan skup znakova trake
- $b \in \Gamma$ znak kojim se označava prazna ćelija
- $\Sigma \subseteq \Gamma \setminus \{b\}$ konačan skup ulaznih znakova
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ parcijalna funkcija prijelaza gdje L i R označavaju pomak lijevo odnosno desno
- $q_0 \in Q$ početno stanje
- $F \subseteq Q$ skup prihvatljivih stanja

Dozvoljava se da je funkcija prijelaza δ nedefinirana za pojedine argumente. Funkcija prijelaza $\delta(q, V) = (p, Z, W)$ određuje da Turingov stroj iz stanja $q \in Q$ čitanjem znaka $V \in \Gamma$ prelazi u stanje p, na traku zapiše znak $Z \in \Gamma$ umjesto znaka V, a glava za čitanje i pisanje miče se u lijevo ili desno ovisno o $W \in \{L, R\}$.

2.2. Rad Turingovog stroja

Turingov stroj se sastoji od trake koja je podijeljena na ćelije koje se nalaze jedna pored druge. Traka ima krajnju lijevu ćeliju dok je beskonačna na desnu stranu. Svaka ćelija sadrži simbol nekog konačnog alfabeta, a budući da se svaki alfabet sastoji od praznog znaka b, pretpostavlja se da je ćelija u koju nikad nije bilo pisano ispunjena upravo tim znakom. Znak b ujedno može poslužiti za brisanje znaka koje je prethodno bilo u polju.

Na početku rada se na traku zapisuje početni izraz, a stroj je u nekom početnom stanju. Turingov stroj nakon čitanja znaka ulazne trake zapiše novi znak na traku. Rad stroja se odvija u uzastopnim taktovima gdje se početna informacija preoblikuje i stroj potencijalno završi s radom. Kretanje i obavljanje operacija pisanja i čitanja se obavlja glavom za čitanje i pisanje koja se može micati u lijevu ili desnu stranu, a taj pomak se može odviti samo za jedno mjesto.



Slika 2.1 Shematski prikaz Turingovog stroja

Obrada informacije odvija se u logičkom bloku, koji se može nalaziti u nekom od konačnog broja stanja Q={q1,q2,...,qN} i on realizira funkciju $\delta(q, V) = (p, Z, W)$ koja svakom paru znakova ulazne dvojke dodjeljuje izlanu trojku.

Čitajući s trake neki znak S_i i ovisno o stanju q u kojem se logički blok nalazio, vrši se upisivanje znaka S_j na traku, promjena stanja logičkog bloka i odluka o pokretanju glave za čitanje i pisanje. Ponašanje stroja određuje funkcija prijelaza δ. Stroj staje kada dođe u neko stanje iz skupa F i tada su na traci zapisani izlazni podaci.

2.3. Primjer beskonačnog rada Turingovog stroja

Pravila prijelaza Turingovog stroja:

```
((q1, b, q1, 1, R)
 (q1, 0, q2, 0, L)
 (q1, 1, q1, 1, R))
```

Pokrenemo li ovaj stroj s konačnim nizom jedinica zapisanih na traci, stroj će zauvijek raditi budući da prema pravilima prijelaza ako pročitamo prazan znak b ili 1 ostajemo u istom (početnom) stanju i idemo u desno neprestano ispisujući 1 na traku. U protivno, ako na traci bude zapisana barem jedna 0 stroj će se zaustaviti tako što je definiran prijelaz u stanje q2 koje je ujedno i završno stanje.

```
---------------------------------------------------------
|   | 1 | 1 | 0 | 1 |   |   |   |   |
---------------------------------------------------------
      ^
     q1
---------------------------------------------------------
|   | 1 | 1 | 0 | 1 |   |   |   |   |
---------------------------------------------------------
          ^
         q1
---------------------------------------------------------
|   | 1 | 1 | 0 | 1 |   |   |   |   |
---------------------------------------------------------
              ^
             q1
---------------------------------------------------------
|   | 1 | 1 | 0 | 1 |   |   |   |   |
---------------------------------------------------------
          ^
         q2
```

Suočivši se s ovim zadatkom bilo bi od velike koristi imati algoritam koji će nam unaprijed reći da li će se Turingov stroj zaustaviti ovisno o nekom početnom zapisu na traci. Želja za takvim algoritmom vodi upravo do Halting problema.

3. Halting problem

Algoritam je konačan slijed dobro definiranih naredbi za ostvarenje nekog zadatka te se može sastojati od petlji koje mogu biti konačnog ili beskonačnog trajanja. Količina posla kojeg treba napraviti ujedno ovisi i o ulaznim podacima. Slijedeće dva primjera prikazuju tu ovisnost i nepredvidivost završetka algoritma u odnosu na ulazne podatke.

Primjer 1.

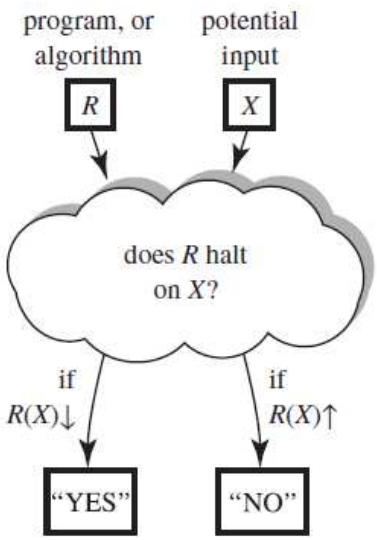
Razmotrimo danu petlju nekog programa:

```
while(x != 1){
        x = x-2; }
```

Zadatak petlje je smanjivati x varijablu za 2 te zaustaviti izvođenje ukoliko x postane jednak 1. S pretpostavkom da je x varijabla pozitivnog integer tipa dozvoljeni ulazi su (1, 2, 3…) te možemo uočiti da će petlja završiti jedino za neparne brojeve dok će se za parne beskonačno izvršavati. Zamislimo li da smo u x pohranili vrijednost 2, u petlji će ta vrijednost opadati za dva i nikada ne poprimiti vrijednost 1 te nastaviti u negativnu stranu.

S namjerom da otkrijemo da li neki kod tj. program s nekim zadanim ulazom završava ili ne možemo osmisliti algoritam koji će odlučivati o tome.

Algoritam za navedeni primjer bi se sastojao od provjere da li je broj paran ili neparan te na temelju toga davao odgovor „DA“ ako program završava sa danim ulazom ili „NE“ ako program nikada neće stati.



Primjer 2.

Razmotrimo novu petlju nekog programa:

```
while( x != 1 ){
        if ( x je paran) x = x/2;
        else x = 3x+1; }
```

Petlja dijeli vrijednost varijable x s 2 ako je x paran broj, a uvećava za više od tri puta ako je neparan broj. Pretpostavimo li da smo varijabli x na početku izvođenja pridijelili vrijednost 7, niz vrijednosti nakon izvođenja petlje bi redom bio: 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5,

16, 8, 4, 2, 1. Kao što vidimo petlja je završila. Točnije, pokrenemo li program s bilo kojim proizvoljnim pozitivnim integer brojem on će završiti. Niz vrijednosti koje poprima x je nepredvidiv i nerijetko poprima visoke vrijednosti prije nego dosegne 1. Iako je petlja godinama testirana na mnoge ulaze i svaki put je završila, ne postoji dokaz da će doista završiti za svaki prirodni broj.

Ovim dvama primjerima je prikazano kako je čak i na jednostavnim algoritmima teško provjeriti osobine i ponašanje u vezi završavanja rada te se otvara tema Halting problema. Problem ima dva ulaza: tekst legalnog programa R u nekom jeziku L i potencijalni ulaz X u program R. Halting problem postavlja pitanje hoće li program R završiti izvođenje ako ga se pokrene s X ulazom te to označavamo $R(X)\downarrow$. Slučaj ako se R izvodi u beskonačnost, tj. divergira, se zapisuje $R(X)\uparrow$.

Halting problem je nerješiv što znači da je nemoguće reći u konačnom vremenu da li će program R završiti s ulazom X. Pretpostavimo li samo da smo pustili program R s ulazom X da se odvija te ako je on završio, jedino što možemo zaključiti je da „DA“, program se zaustavlja s ulazom X, no problem nastaje do kada čekati na taj „DA“ ako izvršavanje predugo nastavi. Ne možemo samo odustati u nekom trenutku i reći da će se program R izvršavati zauvijek s ulazom X jer postoji šansa da ako smo pustili program da se još malo izvodi da bi on došao do kraja.

Definicija Halting problema:

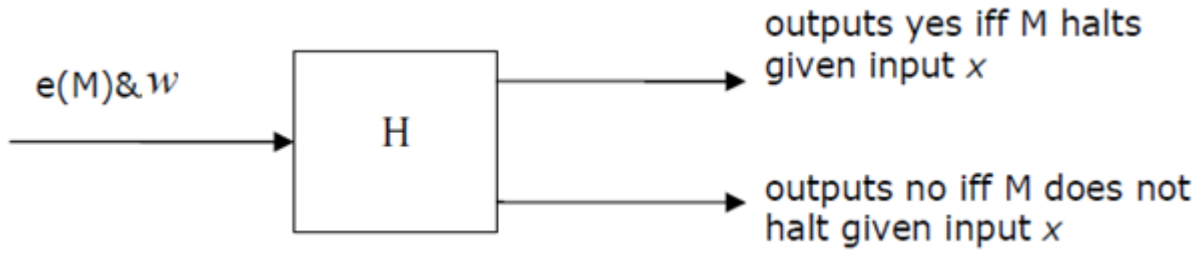
Halting problem je problem odlučivanja da li će neki program ili algoritam uz pripadajući ulaz završiti s izvođenjem ili će se izvoditi zauvijek.

3.1. Nerješivost Halting problema

Teorem:
Halting problem za Turingov stroj je nerješiv.

Dokaz nerješivosti Halting problema:
Zadamo li Turingov stroj $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ i ulaznu vrijednost $w \in \Gamma^*$, pitamo se hoće li se M zaustaviti. Bez smanjenja općenitosti možemo pretpostavljamo da svi Turingovi strojevi u dokazu svoj ulaz kodiraju prirodnim brojevima, prihvaćanje ulaza signaliziraju izlazom 1, a odbacivanje izlazom 0.
Ako je Halting problem rješiv to znači da postoji Turingov stroj H koji za svaki par (e(M), w) daje odgovor da li M završava s ulazom w ili se izvodi zauvijek. Oznaka e(M) predstavlja kodirani zapis Turingovog stroja M.

Želimo dokazati da ne postoji Turingov stroj u jeziku
$L = \{ \langle (M, w) \rangle \mid M \text{ je Turingov stroj koji prihvaća } w \}$ koji rješava Halting problem. Točnije, tvrdimo da ne postoji Turingov stroj H u jeziku L koji dobivši za ulaze Turingov stroj M i ulazni niz w završava u konačnom vremenu odgovorom „DA“ ako Turingov stroj M s ulazom w završi ili „NE“, ako se Turingov stroj R s ulazom w beskonačno izvodi (Slika 3.1).



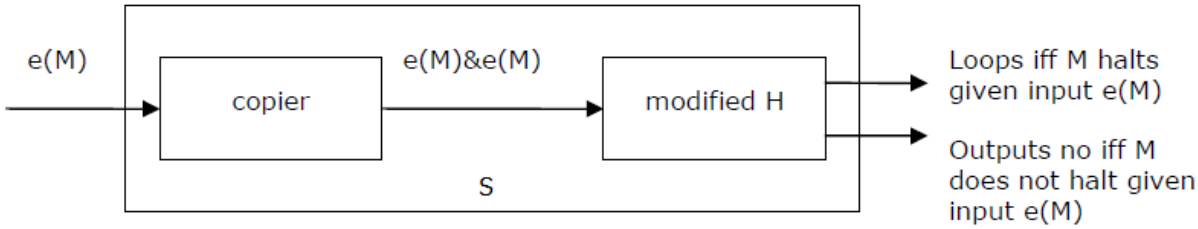
Slika 3.1 Turingov stroj H čije postojanje želimo opovrgnuti

Dokaz po kontradikciji, tj. pretpostavljamo da je Halting problem rješiv te tražimo kontradikciju. Ako pretpostavimo dakle da Turingov stroj H postoji, tada je njime definirana funkcija:

$$H(\ \langle (M, w) \rangle\ ) = 1 \text{ ako je } M(w) = 1$$
$$H(\ \langle (M, w) \rangle\ ) = 0 \text{ ako je } M(w) = 0$$

Takav Turingov stroj ako postoji je također dozvoljeni Turingov stroj jezika L.

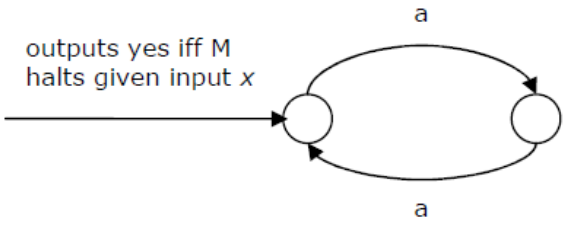
Definirajmo novi Turingov stroj S jezika L. Novostvoreni S možemo shvatiti kao „dijagonalni komplet“ koji na ulaz prima Turingov stroj M, kopira ga i predaje Turingovom stroju H na obadva ulaza. Turingov stroj H, kao što smo prethodno naveli, očekuje dva ulaza odvojena separatorom $e(M)\&w$, a budući da na obadva mjesta predajemo kodirani Turingov stroj M novi ulaz će izgledati $e(M)\&e(M)$ (Slika 3.2).



Slika 3.2 Novi ulazni niz u Turingov stroj H

Turingov stroj S sada čeka da H izvrši svoj posao. Prema pretpostavci da H postoji, on mora završiti s „DA“ ili „NE“ odgovorom budući da smo mu predali Turingov stroj e(M) i posve opravdani ulaz koji je također e(M).

Stroj S će reagirati na završetak Turingovog stroja H na način da će S ući u petlju ako H kaže „DA“ i da će završiti ako H kaže „NE“ (Slika 3.3).



Slika 3.3 Izlazna petlja odgovora "DA"

Nadalje želimo pokazati da je nešto krivo s Turingovim strojem S.

Dakle, činjenica je da za svaki dozvoljeni Turingov stroj M, Turingov stroj S mora ili završiti ili ne. Želimo pokazati da postoji određeni Turingov stroj M za koji S ne može završiti, a ujedno ne može ni nezavršiti. Ovo je dakle kontradikcija pretpostavke da je H moguće konstruirati, te ako to dokažemo dokazali smo nerješivost Halting problema.

Ulazni Turingov stroj M koji čini ovu kontradikciju je upravo sami S.

Da bi pokazali da Turingov stroj S kao ulaz u samog sebe stvara kontradikciju pretpostavimo sljedeće:

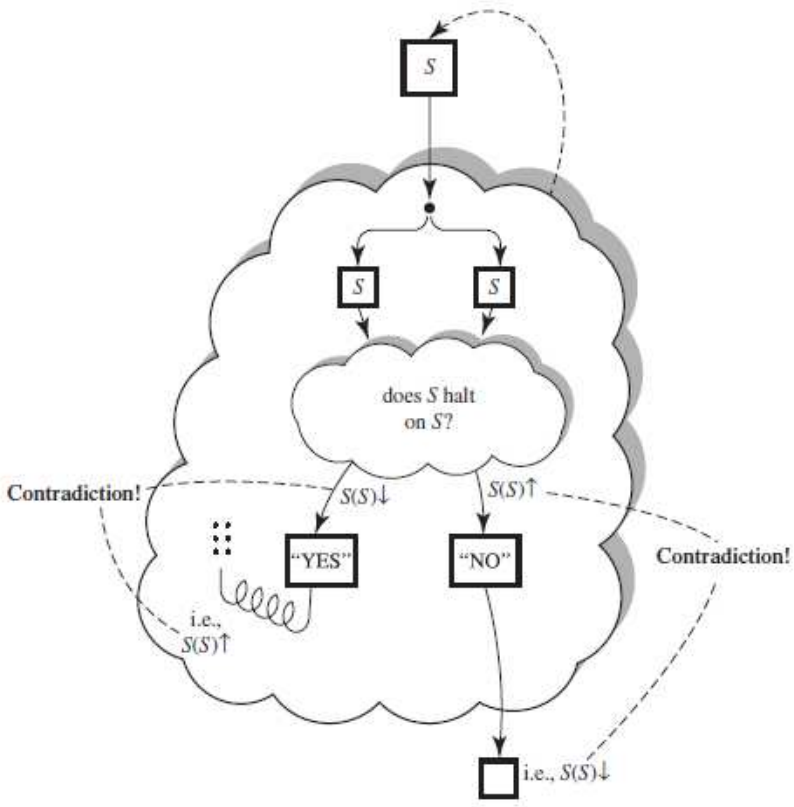
1. **Turingov stroj S kada dobije samog sebe na ulaz završava**

Idemo detaljno kroz pravi tijek događanja u S. Dakle kada S uđe u samog sebe napravi se kopija tog S-a i one se predaju u H kao slijed ulaza. Prema pretpostavci H mora nakon nekog vremena završiti s nekim od odgovora. Ako H izbaci odgovor „DA“ ulazimo u prethodno opisanu beskonačnu petlju te S nikada ne završava i dobili smo kontradikciju.

Nadalje pretpostavimo li da:

2. **Turingov stroj S kada dobije samog sebe na ulaz ne završava**

Promatramo slučaj kada Turingov stroj H vrati odgovor „NE“. Ukoliko se to dogodi S će završiti te je opet došlo do kontradikcije s obzirom na pretpostavku.



Slika 3.4 Kontradikcije u postojanju Turingovog stroja koji rješava Halting problem

Ovim smo dokazali da nešto ne valja s Turingovim strojem S, a budući da je izgrađen svim legalnim potezima jedino što može biti problem je upravo u Turingovom stroju H koji dakle ne može postojati. Dokazom po kontradikciji je dokazana nerješivost Halting problema.

4. Zaključak

Važnost Halting problema leži u činjenici da je bio jedan od prvih problema dokazan kao nerješiv te se nadalje mnogi drugi nerješivi problemi oslanjaju i svode na njegov dokaz.
U prikazanom dokazu moguće je primijetiti da je jedino dokazano da ne postoji Turingov stroj koji rješava Halting problem te se postavlja pitanje da li postoji neki drugi stroj kojim bi to bilo izvedivo? Odgovor na to pitanje daje Church-Turingova teza koja kaže da svako dostižno računalo koliko god snažno bilo je istovjetno Turingovom stroju. Slijedi da Turingov dokaz zajedno s Church-Turingovom tezom doista rezultira nerješivošću Halting problema.
Težina Halting problema leži upravo u zahtjevu da algoritam odluke vrijedi za svaki par algoritma i pripadnog mu ulaza, dok s druge strane krene li se promatrati pojedinačne probleme moglo bi se za neke od njih pronaći prikladan algoritam odluke te se obično oni i izrađuju u dokazne svrhe.

5. Literatura

1. D. Harel, Y. Feldman, „**Algorithmics: The Spirit of Computing**“, Addison-Wesley, 2004., http://books.google.com/books?id=txxLovFWkCUC&hl=hr&source=gbs_navlinks_s
2. B.H.Partee, A.T.Meulen, R.E.Wall, „**Mathematical Methods in Linguistics**“, Kluwer Academic Publishers, 1990., http://books.google.com/books?id=qV7TUuaYcUIC&hl=hr&source=gbs_navlinks_s
3. Z. Manna, „Mathematical Theory of Computation“, Dover Publications, 2003., http://books.google.com/books?id=dwpeNRgjK68C&hl=hr&source=gbs_navlinks_s
4. M.Doko, V.Novaković, „**Izračunljivost i apstraktni strojevi**“, Hrvatski matematički elektronski časopis, http://e.math.unizg.hr/old/izracunljivost/index-print.pdf, [1.11.2009.]
5. Dr.N. Krasnogor, „**Solvability and unsolvability**“, The University of Nottingham, http://www.cs.nott.ac.uk/~nxk/TEACHING/G53COM/G53COMLecture8.pdf, [1.11.2009.]