Formalne metode

u softverskom inženjerstvu

13 Tjuringova mašina

ETFBL 24-25

Dunja Vrbaški

Tjuringova mašina

…	⬜	a	a	b	a	a	b	a	a	⬜	…

$q_0 a \rightarrow q_a \square R$
$q_0 b \rightarrow q_b \square R$
$q_0 \square \rightarrow q_{accept}$

$q'_a a \rightarrow q_L \square L$
$q'_a b \rightarrow q_{reject}$
$q'_a \square \rightarrow q_{accept}$

$q_a a \rightarrow q_a a R$
$q_a b \rightarrow q_a b R$
$q_a \square \rightarrow q'_a \square L$

$q'_b a \rightarrow q_{reject}$
$q'_b b \rightarrow q_L \square L$
$q'_b \square \rightarrow q_{accept}$

$q_b a \rightarrow q_b a R$
$q_b b \rightarrow q_b b R$
$q_b \square \rightarrow q'_b \square L$

$q_L a \rightarrow q_L a L$
$q_L b \rightarrow q_L b L$
$q_L \square \rightarrow q_0 \square R$

Tjuringova mašina

Da li se može napraviti mašina koja simulira rad svih ostalih?

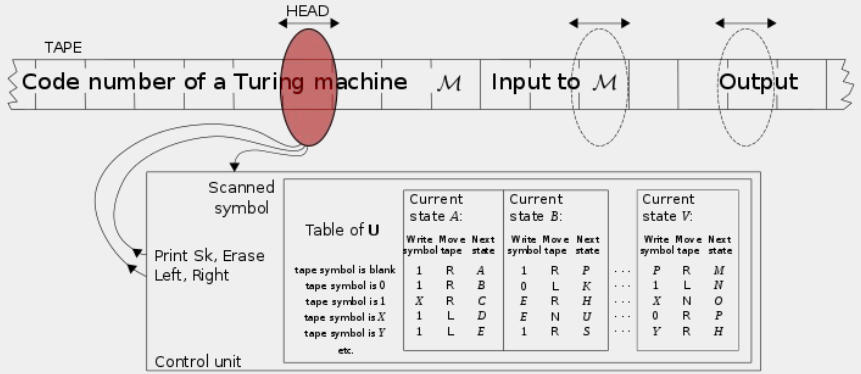
Mašina koja izvršava proizvoljnu mašinu.

Univerzalna TM

Tjuringova mašina koja simulira rad proizvoljne mašine za neki ulaz.

Sa ulaza čita i opis mašine(M) i ulaz za mašinu(w).
Enkodiraju se instrukcije mašine i to predstavlja ulaz za UTM.
U(M, w) = M(w)

Programiranje Tjuringove mašne.



Primer kodiranja

$q_0 \rightarrow 1$, $q_1 \rightarrow 11$, $q_2 \rightarrow 111$,...

$\# \rightarrow 1$, $a \rightarrow 11$, $b \rightarrow 111$, ...

$R \rightarrow 1$, $L \rightarrow 11$

0 - seprator

$(q_3,a) \rightarrow (q_5,b,R)$

11110110111111011101

Primer kodiranja

$(q_3, a) \rightarrow (q_5, b, R)$
$(q_2, a) \rightarrow (q_2, a, L)$

11110110111111011101001110110111011101 1

TM se kodira kao niz simbola.
TM postaje PODATAK.

→ Softver (podatak koji hardver interpretira)

Računar (stored program; instrukcije + podaci u istoj memoriji)

Preteča savremnog računarstva.

Programiranje je moguće!

Šta sve može da se izračuna pomoću TM?

Koje jezike prepoznaje TM?

UTM je alatka u razrašenju ovih pitanja.

Šta može računar da izračuna?

Kada se TM završava:

- TM je završila u finalnom stanju prihvatanja
- TM je završila u finalnom stanju odbacivanja

- TM je završila u nefinalnom stanju

- TM se beskonačno izvršava (petlja)

Koja je definicija/cilj?

- Decider - acc, rej

- Recognizer - acc

- Funkcionalna TM - transformacija podataka, izlaz je dobar

Razmotriti sledeće probleme.

- Izračunati 15. fibonačijev broj
- Izračunati 15 fibonačijevih brojeva
- Proveriti da li je 15 fibonačijev broj
- Izračunati sve fibonačijeve brojeve

Razmotriti sledeće probleme.

- Izračunati 15. fibonačijev broj
- Izračunati 15 fibonačijevih brojeva
- Proveriti da li je 15 fibonačijev broj
- Izračunati sve fibonačijeve brojeve

Fibonačijeve reči (azbuka {0, 1}, konkatenacija)

Ideja: Problemi → problemi sa stringovima

- pripadanje skupu
- nabrajanje skupa

Da li za svaki jezik postoji TM koja može da utvrdi pripadnost jeziku?
Da li za bilo koji ulazni string mašina odgovara sa da ili ne?
Da li druge probleme mozemo svesti na ovakve?

Procedura

Precizan opis postupka

- Izrazi konačne dužine nad konačnom azbukom
- Koraci koji se mogu sprovesti mehanički (bez dodatnih razmatranja)

Imamo uputstvo, ne mora se zaustaviti.
Izvršavanje funkcionalnosti.

Algoritam

Procedura koja se za proizvoljan ulaz završava u konačno mnogo koraka.
Efektivna procedura.

Imamo uputstvo, mora se zaustaviti.
Rešava problem.

Procedura za proveru da li je broj prost - jeste algoritam
Procedura za nalaženje NZD - jeste algoritam
Procedura za pronalaženje korena prirodnog broja - nije algoritam.
Kako kad imamo biblioteke koje vraćaju sqrt?

Church-Turing teza

Za svaki algoritam postoji Tjuringova mašina.

Sve što se efektivno može izračunati mehaničkim sredstvom može se izračunati tjuringovom mašinom.

Ako možeš nešto da izračunaš samo primenjujući određena pravila - postoji TM.

Algoritam je ono što neka Tjuringova mašina izračunava.
Svaki algoritam definiše funkciju koja je tjuring-izračunljiva.

filozofsko-matematički koncept

Church-Turing teza

Za svaki algoritam postoji Tjuringova mašina.
Algoritam je ono što neka Tjuringova mašina izračunava.
Svaki algoritam definiše funkciju koja je tjuring-izračunljiva.

Zašto je teza?

bez formalizma (tehničko + intuitivno)
→ bez dokaza

efektivno izračunljivo - formalno definisan pojam?

- Konačan broj preciznih koraka
- Bez nejasnoća ili intuicije
- Bez neograničenih resursa
- Ono što bi "mehanički računar" (ljudski „kompjuter“) mogao izvesti korak po korak uz papir i olovku

- koristi se na drugim mestima: ako pp da važi teza onda je ...
- snažna podrška da važi, nema kontraprimera
- svi poznati formalni modeli računanja (npr λ–račun) daju iste rezultate kao TM

- Imamo TM: sabiranje, da li je broj prost
- Nemamo TM: Da li TM staje za neki ulaz, da li su dva programa ekvivalentna

Pojam odlučivosti

Vratimo se na jezike.

Jezik TM je skup reči koje mašina prihvata.

TM prihvata/prepoznaje jezik ako na ulazu imamo:

- reč koja pripada jeziku → TM se zaustavlja, prihvata
- reč koja ne pripada jeziku → TM se zaustavlja, odbija ILI se nikad ne zaustavlja

Rekurzivan jezik // odlučivost // staje u oba slucaja // decider

Rekurzivno nabrojiv jezik // prihvatanje // moguća petlja kad reč ne pripada // recognizer

Rekurzivan jezik - reči za koje se TM zaustavlja (postoji algoritam odlučivanja)
Rekurzivno nabrojiv jezik - reči koje TM “nabraja” (postoji procedura koja nabraja)

Svaki rekurzivan je i rekurzivno nabrojiv? DA

- generišemo sve reči nad azbukom
- algoritmom odlučivanja utvrđujemo da li pripadaju jeziku

→ dobijamo nabrajanje

Svaki rekurzivno nabrojiv ujedno i rekurzivan? NE

- izaberemo proizvoljnu reč
- krenemo da nabrajamo sve reči
- ako reč pripada jeziku - naići ćemo na nju
- ako reč ne pripada jeziku - možemo dugo i uzaludno čekati da se pojavi (bez uverenja da li ćemo dočekati)

ovo je intuitivno. formalan dokaz je komplikovaniji

Da li je svaki rekurzivno nabrojiv zaista i jezik tipa 0?
Da li za proizvoljnu proceduru možemo naći odgovarajuću gramatiku?

PROBLEM:

pojam procedure - intuitivno definisan i prihvaćen
gramatika - metematički precizno definisan pojam

Chomsky uvodi jer pokušava da formalizuje govorne jezike.

Praktično za nas:
- regularni jezici
- CFG jezici
- izračunljivost → složenost

Jezici tipa 1, 2, 3

proizvoljna kontekstno osetljiva gramatika + proizvoljna reč
Imamo algoritam, mašinu, koji odlučuje da li reč pripada jeziku.

→

Problem pripadanja reči za klasu CS jezika je algoritamski odlučiv.
(ostali, jednostavniji su podskup CS)

To što je odlučiv ne znači i da je “lak” za rešavanje.

Jezici tipa 0, gramatika bez ograničenja: Da li algoritamski odlučiv?

- DA: naći algoritam
- NE: pokazati da algoritam ne postoji

Halting problem za Tjuringove mašine je algoritamski neodlučiv.

Imamo:
- proizvoljnu proceduru (program)
- proizvoljnu reč (ulaz)

Da li će se zaustaviti?

Ne postoji algoritam koji će za proizvoljan program i proizvoljan ulaz uvek odlučiti da li će se program zaustaviti (halt) ili ne.

imamo prihvatanje, ali ne i odlučivanje

tvrđenje i dokaz samo koristeći tjuringove mašine, bez church teze i intuitivnih pojmova
→ još jedan prilog da teza važi

Dokaz (ideja)

Pretpostavimo da postoji takva funkcija halts koja za proizvoljnu funkciju vraća da li se ta funkcija zaustavlja ili ne.

- Da li postoji neka funkcija koja radi suprotno od onog što očekujemo kao rezultat funkcije halts?
- Da li mozemo da primenimo na samu sebe?

```
function halts(f)
     return T or F

function g()
    if halts(g):
        loop()
```