

# Teoretične osnove računalništva

## Zapiski predavanj 2010/2011

28. februar 2011



This work is licensed under a Creative Commons  
Attribution-NonCommercial-ShareAlike 3.0  
Unported License

# Kazalo

<b>1</b>	<b>Turingov Stroj</b>	<b>2</b>
1.1	Zgodovina . . . . .	2
1.2	Turingovi stroji . . . . .	3
1.2.1	Trenutni opis . . . . .	3
1.2.2	Relacija $\vdash$ . . . . .	3
1.2.3	Tranzitivna ovojnica $\vdash^*$ relacije $\vdash$ . . . . .	3
1.3	Jezik Turingovega stroja . . . . .	4
1.3.1	Ugotavljanje pripadnosti besed Turingovemu jeziku . . . . .	4
1.3.2	Turingov stroj kot računalnik funkcij . . . . .	5
1.3.3	Lažja konstrukcija Turingovih strojev . . . . .	5

# Poglavje 1

## Turingov Stroj

### 1.1 Zgodovina

Leta 1900 je Nemški matematik David Hilbert objavil seznam triidvajsetih nerešenih problemov v matematiki. Eden izmed Hilbertovih problemov (deseti po vrsti), je vprašanje, ali obstaja postopek, po katerem ugotovimo rešljivost poljubne Diofantske enačbe – torej, ali lahko ugotovimo, če ima polinom s celoštevilskimi koeficienti  $P(x_1, x_2, \dots, x_n) = 0$ , celoštevilsko rešitev. Kljub temu, da je Emil Post že leta 1944 slutil, da je problem nerešljiv, je to dokončno dokazal rus Jurij Matijaševič šele leta 1970 v svojem doktorskem delu. Med reševanjem problema pa so se matematiki že prej začeli ukvarjati s formalizacijo pojma postopka oz. algoritma. Intuitivna definicija tega se glasi nekako tako:

**Def.:** Algoritem je zaporedje ukazov, s katerimi se v končnem številu korakov opravi neka naloga.

Pri tem pa ostaja še kar nekaj odprtih vprašanj, npr.:

- Kakšni naj bodo ukazi?
  - Osnovni - algoritem ima veliko korakov
  - Kompleksni - prezapleteni ukazi so že sami algoritmi
- Koliko ukazov naj bo?
  - Končno - ali je s končno množico res mogoče rešiti vsako nalogo?
  - Neskončno - kakšen izvajalec ukazov je sposoben izvršiti neskončno različnih ukazov?
- So ukazi zvezni ali diskretni?
- V kakšnem pomnilniku so ukazi shranjeni?
  - Končnem - ali s končnim zaporedjem ukazov res lahko mogoče rešimo vsako nalogo?
  - Neskončnem -

Nekateri zgodnji poskusi formalizacije pojma algoritma so:

- GK (Kurt Gödel, Stephen Kleene)
- HG (Jacques Herbrand, Kurt Gödel)
- Produkcijski sistem (Emil Post),
- Lambda račun (Alonso Church, 1936)
- Turingov stroj (Alan Turing, 1936)

## 1.2 Turingovi stroji

Turingov stroj se je uveljavil kot uporaben in preprost model računanja, ki zna izračunati vse kar se izračunati da (pod pogojem, da Church-Turingova teza drži). Alan Turing je svoj stroj izpeljal iz razmišljanja o tem, kako človek rešuje miselne probleme na papir. Pri tem je izbral tri sestavne dele:

- Nadzorno enoto (glava)
- Čitalno okno (roka in vid)
- Trak (papir)

V postopku formalizacije, pa je zaradi večje preprostosti, zahteval še, da je stroj sestavljen iz končno mnogo elementov, ter da deluje v diskretnih korakih.

**Def.:** Turingov stroj je definiran kot sedmerka  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, B, F \rangle$ , kjer je:

- $Q$  končna množica stanj
- $\Sigma$  končna množica vhodnih simbolov,  $Q \cap \Sigma = \emptyset$
- $\Gamma$  končna množica tračnih simbolov,  $\Sigma \subset \Gamma$
- $\delta$  funkcija prehodov:  $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, D\}$ ,  
kjer  $L$  in  $D$  označujeta premik levo ali desno
- $q_0$  začetno stanje,  $q_0 \in Q$
- $B$  prazen simbol,  $B \in \Gamma$
- $F$  množica končnih stanj,  $F \subseteq Q$

Stroj deluje tako, da v vsakem koraku opravi naslednje:

- preide v neko stanje
- zapiše nov simbol v celico, ki je pod oknom
- okno premakne eno celico levo ali desno

### 1.2.1 Trenutni opis

**Def.:**  $TO = \Gamma^* \times Q \times \Gamma^*$  je množica vseh trenutnih opisov.

Nek trenutni opis  $(\alpha_1, q, \alpha_2)$ , ali krajše  $\alpha_1 q \alpha_2$  opisuje konfiguracijo Turingovega stroja.

Iz  $\alpha_1$  in  $\alpha_2$ , lahko razberemo:

- če je  $\alpha_1 = \varepsilon$ , je okno skrajno levo
- če je  $\alpha_2 = \varepsilon$ , je okno nad  $B$  in so naprej sami  $B$ -ji

### 1.2.2 Relacija $\vdash$

**Def.:** Če sta  $u, v$  trenutna opisa iz množice  $TO$ , ter  $v$  neposredno sledi iz  $u$  v enem koraku Turingovega stroja, tedaj pišemo  $u \vdash v$ .

Naj bo  $x_1 \dots x_{i-1} q x_i \dots x_n$  trenutni opis:

- če je  $\delta(q, x_i) = (p, Y, D)$ :  
 $x_1 \dots x_{i-1} q x_i \dots x_n \vdash x_1 \dots x_{i-1} Y p x_{i+1} \dots x_n$
- če je  $\delta(q, x_i) = (p, Y, L)$ :  
\* če je okno na robu ( $i = 1$ ), se Turingov stroj ustavi, ker je trak na levi omejen.  
\* če okno ni na robu ( $i > 1$ ), potem:  $x_1 \dots x_{i-2} x_{i-1} q x_i \dots x_n \vdash x_1 \dots x_{i-2} p x_{i-1} Y x_{i+1} \dots x_n$

### 1.2.3 Tranzitivna ovojnica $\vdash^*$ relacije $\vdash$

**Def.:**  $u \vdash^* v$ , če obstaja tako zaporedje  $x_i, (i \in [0, 1, \dots, k], k \geq 0)$ , da velja  $u = x_0, v = x_k$  in  $x_0 \vdash x_1 \wedge x_1 \vdash x_2 \wedge \dots \wedge x_{k-1} \vdash x_k$

Torej, trenutni opis  $v$  sledi iz  $u$ , v  $k$  korakih Turingovega stroja.

## 1.3 Jezik Turingovega stroja

**Def.:** Jezik Turingovega stroja je definiran kot:

$$L(M) = \{w \mid w \in \Sigma^* \wedge q_0 w \vdash^* w_1 q w_2 \wedge w_1, w_2 \in \Gamma^* \wedge q \in F\}$$

Z besedami to pomeni, da je  $L(M)$  množica besed  $w \in \Sigma^*$ , ki če jih damo na vhod stroju  $M$ , povzročijo, da se stroj  $M$  v končno mnogo korakov znajde v končnem stanju.

**Def.:** Jezik  $L$  je Turingov jezik, če obstaja Turingov stroj  $M$ , tak, da je  $L = L(M)$ .

### 1.3.1 Ugotavljanje pripadnosti besed Turingovemu jeziku

Pri vprašanju ali je neka beseda v jeziku, Turingove jezike ločimo na:

- Odločljive - obstaja algoritem, s katerim se lahko za poljubno besedo odločimo, ali pripada jeziku.
- Neodločljive - v splošnem ni algoritma, ki bi za poljubno vhodno besedo z DA ali NE odgovoril na vprašanje pripadnosti.
  - če je odgovor DA, to ugotovimo v nekem končnem številu korakov.
  - če je odgovor NE, pa ni nujno, da se bo stroj kdaj ustavil.

#### Primeri:

**Primer 1:** Zapiši Turingov stroj, ki sprejema jezik  $L = \{0^n 1^n \mid n \geq 1\}$

Osnovna ideja:

- $0^n 1^n$  - vhodna beseda
- $X0^{n-1}1^n$  - zamenjamo levo 0 z X
- $X0^{n-1}Y1^{n-1}$  - premaknemo se desno do najbolj leve 1 in jo zamenjamo z Y
- $XX0^{n-2}Y1^{n-1}$
- $XX0^{n-2}YY1^{n-2}$  - ponovimo in vidimo, da bomo niz sprejeli, če je prave oblike.

Turingov stroj zapišemo kot  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, B, F \rangle$ :

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, B, X, Y\}$
- $F = \{q_4\}$
- $\delta$  bomo definirali s tabelo

Pomen stanj:

- $q_0$  - začetno stanje in stanje pred zamenjavo 0 z X
- $q_1$  - premikanje desno do 1
- $q_2$  - zamenjava 1 z Y in premikanje levo do X
- $q_3$  - najde X in se premik desno
- $q_4$  - končno stanje

Tabela prehajanja stanj:

	0	1	B	X	Y
$x_0$	$\langle q_1, X, D \rangle$	–	–	$\langle q_3, Y, D \rangle$	–
$x_1$	$\langle q_1, 0, D \rangle$	$\langle q_2, Y, L \rangle$	–	$\langle q_1, Y, D \rangle$	–
$x_2$	$\langle q_2, 0, D \rangle$	–	$\langle q_0, X, D \rangle$	$\langle q_2, Y, L \rangle$	–
$x_3$	–	–	–	$\langle q_3, Y, D \rangle$	$\langle q_4, B, D \rangle$
$x_4$	–	–	–	–	–

Izvajanje stroja s trenutnimi opisi:

$$q_0 0011 \vdash Xq_1 011 \vdash X0q_1 11 \vdash Xq_2 0Y1 \vdash \dots$$

### 1.3.2 Turingov stroj kot računalnik funkcij

Imamo Turingov stroj, ki ima na traku neko število ničel, ki predstavljajo pozitivna naravna števila, ločena z enicami:

$$0^{i_1} 1 0^{i_2} 1 \dots 1 0^{i_k}$$

Recimo, da se stroj po nekem številu korakov ustavi in ima na traku skupino ničel  $0^m$ , na levi in desni strani skupine pa same  $B$ -je. S tem je stroj lahko izračunal neko funkcijo

$$f^{(k)} : \mathbb{N}_+^k \rightarrow \mathbb{N}_+ \text{ oz. } f(i_1, i_2, \dots, i_k) = m$$

Funkcija  $f$  ni nujno definirana za vsako  $k$ -terico iz  $\mathbb{N}_+^k$ , torej je parcialna funkcija, kadar pa je definirana povsod, pravimo da je totalna. Stroj se pri nedefiniranih  $k$ -tericah pač na neki točki ustavi in pri tem na traku ne pusti le ene skupine ničel. Isti turingov stroj hkrati računa več funkcij:  $f^{(1)}, f^{(2)}, \dots, f^{(k)}$ .

#### Parcialna rekurzivna funkcija

**Def.:** Vsaka funkcija  $f^{(k)} : \mathbb{N}_+^k \rightarrow \mathbb{N}$ , ki jo lahko izračuna nek Turingov stroj, je parcialna rekurzivna funkcija. Če je  $f^{(k)}$  definirana za vse  $k$ -terice, jo imenujemo totalna rekurzivna funkcija (včasih samo rekurzivna funkcija)

Vse običajne aritmetične funkcije (na  $\mathbb{N}^k$ ) so parcialne ali celo totalne rekurzivne funkcije. V nadaljevanju si bomo pogledali nekaj primerov, tu jih le nekaj naštejmo:  $+$ ,  $*$ ,  $n!$ ,  $2^n$ ,  $\lceil \log(n) \rceil$ ,  $m^n$ ,  $\dots$

#### Primeri:

**Primer 1:** Ali je  $f(m, n) = m + n$  rekurzivna?

Skica stroja, ki računa  $m + n$ :

- $0^m 10^m$  - vhodna beseda
- $B0^{m-1}10^m$  - izbriši prvo ničlo
- $B0^{m+n}$  - premakni se do 1 in jo zamenjaj z 0

**Primer 2:** Ali je  $f(m, n) = m * n$  parcialno rekurzivna?

Skica stroja, ki računa  $m * n$ :

- $0^m 10^n$  - vhodna beseda
- $0^m 10^n 1$  - premakni se na konec in zapiši 1 (ločnica za rezultat)
- $B0^{m-1}10^n 1$  - premakni se na začetek in izbriši 0
- $B0^{m-1}10^m 10^n$  - prekopiraj  $n$  ničel za ločnico (in ničle)
- $B^m 10^m 10^{m*n}$  - ponavljaj tadv koraka, dokler ni več ničel pred prvo 1
- $B^{m+n+2}0^{m*n}$  - izbriši del, ki ne spada v rezultat

### 1.3.3 Lažja konstrukcija Turingovih strojev

Obstaja nekaj tehnik, ki poenostavijo in pohitijo sestavljanje Turingovih strojev.

Nadzorna enota kot pomnilnik

Večsledni trak

Prestavljanje vsebine traku

Podprogrami