Izračunljivost i složenost

Andrej Ivašković

12. decembar 2016.

§1. Teorija izračunljivosti

- Zanima nas *formalizacija* pojma programa.

- Funkcije kao koncept u programskim jezicima odgovaraju, matematički, *parcijalnim funkcijama*. Na primer, u C-ovskom jeziku koji bi mogao da operiše sa celim brojevima (umesto $\left\{-2^{31}, \ldots, 2^{31}-1\right\}$):

```
int cond_id(int x)
{
  while (x < 0) {}
  return x;
}
```

  odgovara parcijalnoj funkciji $f : \mathbb{Z} \rightharpoonup \mathbb{Z}$ koja zadovoljava:

$$f(x) = \begin{cases} x & x \geqslant 0 \\ \uparrow & x < 0 \end{cases}$$

- **Teorija izračunljivosti** razmatra za koje parcijalne funkcije postoji program koji ih računa, odnosno koje funkcije su **izračunljive**.

- Mogući apstraktni modeli izračunljivosti:

  - parcijalne rekurzivne funkcije
  - lambda račun (*Alonzo Čerč*)
  - Tjuringove mašine (*Alan Tjuring*)
  - registarske mašine (*Marvin Minski*)
  - programski jezici (uz formalno definisanu semantiku)

- **Čerč-Tjuringova teza.** Svi navedeni apstraktni modeli izračunljivosti su ekvivalentni i odgovaraju intuitivnom razumevanju pojma izračunljivosti.

- Tjuringova mašina odgovara jednom programu i radnoj memoriji, sastoji se od:

  - **skupa stanja** ("režimi rada"), uz specijalna stanja `acc` i `rej`
  - **tablice akcija** ("program"), gde akcije mogu da promene trenutno stanje, promene simbol na trenutnoj poziciji trake i pomere traku, u zavisnosti od aktuelnog stanja i simbola na traci
  - **trake** (beskonačne "memorije"), simboli su `0`, `1`, `_`, `>` (traka počinje sa `>`, svi simboli počev od nekog su `_`)

- **Konfiguracija Tjuringove mašine:** (s, l, r), gde je s stanje u kom se Tjuringova mašina nalazi, l je niz simbola od početka trake do aktuelnog, a r niz simbola posle aktuelnog (do beskonačnog ponavljanja `_`).

- Logična definicija **promene** konfiguracije, $c_1 \rightarrow c_2$. Ovo predstavlja "jedan korak", a "više koraka" se zapisuje $c_1 \rightarrow^* c_n$ ako $c_1 \rightarrow c_2$, $c_2 \rightarrow c_3 \ldots$, $c_{n-1} \rightarrow c_n$.

- Kažemo da konfiguracija c dovodi do **prekida** ukoliko postoji konfiguracija $c' = (s, l, r)$ (gde $s \in \{\mathtt{acc}, \mathtt{rej}\}$) takva da $c \rightarrow^* c'$.

- Početna konfiguracija: zahteva inicijalne simbole na traci i početno stanje.

- **Primer.** Šta radi naredna Tjuringova mašina:

  - skup stanja je $S = \{\mathtt{mov}, \mathtt{acc}, \mathtt{rej}\}$, početno je `mov`
  - akcije su zadate tablicom:

	`>`	`_`	`0`	`1`
`mov`	$(\mathtt{>}, \Rightarrow, \mathtt{mov})$	$(\mathtt{0}, \Downarrow, \mathtt{acc})$	$(\mathtt{0}, \Rightarrow, \mathtt{mov})$	$(\mathtt{1}, \Rightarrow, \mathtt{mov})$

  ukoliko je početno stanje trake `>101001___`…?

- Postoji bijekcija između $\mathbb{N}_0$ i skupa svih konfiguracija svih Tjuringovih mašina. Drugim rečima, postoji *kodiranje* Tjuringovih mašina (koje je izračunljivo)!

- Štaviše, $\mathbb{N}_0$ je dovoljan skup za kodiranje svega sa čim se susrećemo u "klasičnom" računarstvu.

- Parcijalna funkcija $f : \mathbb{N}_0 \rightharpoonup \mathbb{N}_0$ je **izračunljiva** ukoliko postoji Tjuringova mašina sa startnim stanjem s takva da, ako je a kodiranje $n \in \mathbb{N}_0$, konfiguracija $c = (s, >, a)$ se prekida u c' ($c \rightarrow^* c'$) ako i samo ako $f(n)\downarrow$, a u tom slučaju konačno stanje trake kodira $f(n)$.

- Skup $A \subseteq \mathbb{N}_0$ je **odlučiv** ukoliko postoji Tjuringova mašina koja uvek dovodi do prekida, pri čemu je za ulazne trake koje kodiraju $n \in A$ finalno stanje `acc`, a u suprotnom `rej`.

- **Entscheidungsproblem (Hilbert).** Za dat iskaz (izraženom u jeziku logike prvog reda) odrediti da li je tačan: ako jeste, proizvesti dokaz; ako nije, dati kontraprimer.

- **Halting problem.** Da li postoji Tjuringova mašina koja, za dat kod e neke Tjuringove mašine M i polaznu konfiguraciju c, odlučuje da li dolazi do prekida? $\Longrightarrow$ NE!

- **Skica dokaza.**

- Napredniji rezultati teorije izračunljivosti nam kažu da ništa "zanimljivo" nije izračunljivo!

§2. Klase složenosti

- Možemo li da kažemo nešto o *broju konfiguracija* kroz koje prolazi jedna Tjuringova masina (ukoliko dodje do prekida)?

- Kažemo da Tjuringova mašina ima **vremensku složenost** $O(f(n))$ ukoliko za bilo koju ulaznu traku veličine n ona prolazi kroz $O(f(n))$ konfiguracija u najgorem slučaju (*ukoliko dođe do prekida*).

- Tada za odgovarajući problem odlučivanja (odnosno skup za koji se traži algoritam odlučivanja) definišemo da je u skupu **TIME** $(f(n))$.

- Skup svih problema odlučivanja za koje postoji polinom $p(n)$ za koji je problem u **TIME** $(p(n))$ označavamo sa **P**. Drugi način da ovo zapišemo je takođe:
$$\mathbf{P} = \bigcup_{k=0}^{\infty} \mathbf{TIME}\left(n^k\right)$$

- Primeri problema u **P**:

  - odlučivanje da li je zbir elemenata konačnog niza jednak nuli;
  - odlučivanje da li usmeren graf ima cikluse;
  - odlučivanje da li je prirodan broj prost.

- Nije teško proširiti ovu priču sa problema odlučivanja i na izračunljive parcijalne funkcije, ali to su uglavnom tehnički detalji.

- Postoji bijekcija između Tjuringovih mašina sa jednom trakom i Tjuringovih mašina sa dve trake, gde je jedna nepromenljiva (ulazna), a druga je "radna memorija". Na osnovu veličine radne memorije se definiše **prostorna složenost** $O(f(n))$.

- Po analogiji imamo i skup **SPACE** $(f(n))$, kao i narednu klasu složenosti:

$$\mathbf{PSPACE} = \bigcup_{k=0}^{\infty} \mathbf{SPACE}\left(n^k\right)$$

- Ukoliko umesto jasno određenog koraka za svaki par simbola i stanja imamo skup mogućih akcija, tada je reč o **nedeterminističkoj Tjuringovoj mašini**. U ovom slučaju kažemo da je **vremenska složenost** reda $O(f(n))$ ukoliko postoji *bar jedan* niz promena konfiguracija takav da u $O(f(n))$ koraka dolazi do prekida.

- Po analogiji definišemo **NTIME** $(f(n))$ i:

$$\mathbf{NP} = \bigcup_{k=0}^{\infty} \mathbf{NTIME}\left(n^k\right)$$

- Jedna moguća ekvivalentna interpretacija **NP** je takođe i postojanje *verifikacije* u polinomskom vremenu.

- Jasno je da je $\mathbf{P} \subseteq \mathbf{NP}$, ali je otvoren problem $\mathbf{P} = \mathbf{NP}$! Jedan od *milenijumskih problema.*

- Za $\mathbf{P} \neq \mathbf{NP}$ je dovoljno pokazati da postoji problem u **NP** koji nije u **P**.

- Za data dva skupa $A \subseteq \mathbb{N}_0$ i $B \subseteq \mathbb{N}_0$, zovemo izračunljivu $f : \mathbb{N}_0 \to \mathbb{N}_0$ **redukcijom** iz A u B ukoliko za sve $x \in \mathbb{N}_0$, $f(x) \in B$ ako i samo ako $x \in A$.

- Ukoliko je f izračunljiva u polinomskom vremenu, tada je reč o **polinomskoj redukciji**.

- Kažemo da je skup A **NP-težak** ukoliko za svaki $X \in \mathbf{NP}$ postoji polinomska redukcija iz X u A.

- A je **NP-kompletan** ukoliko je u **NP** i takođe je **NP**-težak.

- **Kukova teorema.** Problem Sat (da li je logička formula zadovoljiva) je **NP**-kompletan.

- Na narednom predavanju ćemo pokriti nekoliko bitnih **NP**-kompletnih problema! Oni su svi među sobom *ekvivalentni.*

- Ukoliko bi se pokazalo da je neki **NP**-kompletan problem u **P**, tada bi važilo $\mathbf{P} = \mathbf{NP}$. Važi i obratno.

- Hijerarhija: