I Z R A Č U N L J I V O S T

⋙ *p r e d a v a n j a & v j e ž b e* ⋘

Slajdovi su *pomoć u nastavi*, nisu za samostalno učenje!

Vedran Čačić <veky@math.hr>
[konzultacije: ponedjeljkom, 12–14 uz najavu]

PMF–Matematički odsjek
https://web.math.pmf.unizg.hr/~veky/izr
[ili samo guglajte „izračunljivost”:]

akademska godina 2023./2024.
zadnja promjena: 20. studenoga 2023.

Literatura i pravila polaganja

- V. Čačić, *Komputonomikon*, udžbenik, 2022.
- V. Čačić, *Komputrivij*, zbirka zadataka (ZZ), 2022.
- M. Vuković, *Izračunljivost*, skripta, PMF–MO, 2015.
- J. Shoenfield, *Recursion theory*, Springer-Verlag, 1993.

Elementi ocjenjivanja

- kolokvij (15 bodova), prvi dio gradiva, & teorijska pitanja
- pisani ispit (85 bodova), (+ kolokvij) ≥ 40 za prolaz/usmeni
- usmeni ispit: teorija, završna ocjena (gledajući i bodove)

Prije izlaska na ispit možete izraditi seminar: javite se za temu do zadnjeg predavanja. Tada usmeno odgovarate samo tu temu.

Što je izračunljivost?

Ukratko, proučavanje algoritama kao matematičkih objekata.

Zašto nam to treba? Prvenstveno da bismo dokazali da za dani problem *ne postoji* algoritam. Primjerice, da bismo dokazali ...

Primjer (Churchov teorem, Matematička logika)

Logika prvog reda je neodlučiva: ne postoji algoritam koji za proizvoljnu formulu prvog reda određuje je li valjana.

Još neki primjeri problema za koje ne postoje algoritmi:

- (Diofantske jednadžbe) Neka je $p(x_1, \ldots, x_n)$ polinom s cjelobrojnim koeficijentima. Ima li p nultočku u $\mathbb{Z}^n$?
- (Post) Neka je R konačna binarna relacija nad $\{0,1\}^*$. Postoje li $a_1 \mathrel{R} b_1, \ldots, a_n \mathrel{R} b_n$ za koje je $a_1 \cdots a_n = b_1 \cdots b_n$?

Za velike potklase gornjih problema su poznati algoritmi, no za općenite probleme znamo da ne postoje.

Algoritmi kroz povijest

- Babilon, –17. stoljeće: procedura za rekonstrukciju dvaju brojeva iz njihove razlike i umnoška
- Grčka, –3. stoljeće: Euklidov algoritam za određivanje najveće zajedničke mjere dviju dužina
- Perzija, 9. stoljeće: Al-Ḫorezmi zapisuje metode računanja s arapskim brojkama (učimo ih u osnovnoj školi)
- Italija, 16. stoljeće: Ferrarijeva metoda za rješavanje algebarskih jednadžbi četvrtog stupnja
- Indija, 21. stoljeće: AKS, polinomni test za prim-brojeve

Usprkos vrlo različitim matematičkim formalizacijama u kojima su formulirani, te različitim tipovima podataka s kojima su radili, svima njima je nešto zajedničko. Što?

Intuitivni opis pojmova

Izračunavanje je postupak kojim iz nekih zadanih objekata (*ulaznih podataka*), fiksiranim skupom mehaničkih pravila (*algoritmom*) dobivamo krajnji rezultat (*izlazni podatak*).

Ako za izračunavanje želimo naglasiti algoritam A i ulazne podatke x, zovemo ga *A-izračunavanje s x*.

Pretpostavljamo točno jedan izlazni podatak izračunavanja. Izračunavanje s l izlaznih podataka modeliramo pomoću l nezavisnih izračunavanja s istim ulaznim podacima.

S druge strane dozvoljavamo jedan ili više ulaznih podataka (*mjesnost*), ali *konačno* mnogo i *fiksno* za zadani algoritam.

Nije nužno da za sve ulazne podatke algoritam daje rezultat — ponekad može raditi beskonačno dugo bez zaustavljanja.

Brojevne funkcije

Podaci su nam, od sad pa dok ne kažemo drugačije, isključivo prirodni brojevi s nulom. Označavamo $\mathbb{N}_+ := \mathbb{N} \setminus \{0\}$. Druge objekte *kodiramo* prirodnim brojevima.

Dakle, algoritmima računamo (neke) funkcije f čija domena $\mathcal{D}_f$ je podskup od $\mathbb{N}^k$ za neki $k \in \mathbb{N}_+$, a kodomena im je $\mathbb{N}$. Ubuduće podrazumijevamo funkcije takvog oblika (*brojevne*).

Broj k zovemo *mjesnošću* funkcije f, kažemo da je f k-*mjesna* funkcija, i pišemo f^k ako to želimo istaknuti. Sāmu k-torku $(x_1, \ldots, x_k) \in \mathbb{N}^k$ skraćeno pišemo $\vec{x}$ ili $\vec{x}^k$.

Kažemo da je funkcija f^k *totalna* ako je $\mathcal{D}_f = \mathbb{N}^k$.
Prazna funkcija $\otimes$ ima domenu $\varnothing$ (nije definirana nigdje).
Svaka neprazna funkcija ima jedinstvenu mjesnost.
Pravit ćemo se da i prazne funkcije imaju mjesnost: $\otimes^k$, $k \in \mathbb{N}$.

Izračunavanje funkcija

Definicija

Kažemo da algoritam $\mathcal{A}$ mjesnosti k *računa* funkciju f^k ako za sve $\vec{x} \in \mathbb{N}^k$ vrijedi:

- Ako je $\vec{x} \in \mathcal{D}_f$, tada $\mathcal{A}$-izračunavanje s $\vec{x}$ stane, te mu je izlazni podatak jednak $f(\vec{x})$.
- Ako $\vec{x} \notin \mathcal{D}_f$, tada $\mathcal{A}$-izračunavanje s $\vec{x}$ nikada ne stane.

Funkcija f je *izračunljiva* ako postoji algoritam koji je računa.

To nije „prava” definicija jer je pojam algoritma još nejasan. Formalizacijom pojma algoritma (i izračunavanja) dobit ćemo i formalnu definiciju izračunljive funkcije: npr. funkcija je *RAM*-izračunljiva ako postoji *RAM-algoritam* (taj pojam ćemo precizno definirati) koji je računa.

Nekonstruktivna egzistencija algoritma

Trebamo biti oprezni: „dokazati da je funkcija izračunljiva” ne znači nužno „naći algoritam za nju”.

[Ipak, najčešće ćemo doista konstruirati algoritme.]

Primjer

$$f(n) := \begin{cases} 1 & \text{ako postoji } n \text{ uzastopnih peticā u decimalama } \sqrt{2} \\ 0 & \text{inače} \end{cases}$$

jest izračunljiva, iako ne znamo napisati algoritam za nju.

Ključno je da je f padajuća, a svaka padajuća funkcija na prirodnim brojevima je izračunljiva. (To ćemo dokazati.)

Funkcije poput ove u primjeru, čija slika je podskup od $\{0,1\}$ i provjeravaju određena svojstva, su česte. Njima modeliramo ...

Relacije

Za $k \in \mathbb{N}_+$, k-*mjesna relacija* R je bilo koji podskup od $\mathbb{N}^k$.

Ubuduće podrazumijevamo relacije takvog oblika (*brojevne*).

Kao i za funkcije, k zovemo *mjesnost* relacije R, i pišemo R^k.

Svaka neprazna relacija ima jedinstvenu mjesnost.

Činjenicu $\vec{x} \in R$ pišemo još i $R(\vec{x})$. Dvomjesne ($k = 2$) relacije (i funkcije, npr. aritmetičke operacije) često pišemo infiksno.

Poistovjećujemo skup $\mathbb{N}^1$ sa skupom $\mathbb{N}$: jednomjesne ($k = 1$) relacije gledamo kao podskupove skupa prirodnih brojeva.

U teoriji skupova veza između relacija i funkcija glasi: funkcije su specijalne relacije (one s funkcijskim svojstvom).

Algoritamski, prirodna veza ide u suprotnom smjeru: relacije su specijalne funkcije (one s kodomenom $\mathrm{bool} = \{0, 1\}$).

Relacije s funkcijskim svojstvom i njihovu vezu s funkcijama proučavat ćemo puno kasnije, kad budemo govorili o grafovima.

Izračunljivost relacijā

Za relaciju R^k, s χ_R^k označavamo njenu *karakterističnu funkciju*

$$\chi_R(\vec{x}) := \begin{cases} 1, & \text{ako vrijedi } R(\vec{x}) \\ 0, & \text{inače} \end{cases}.$$

Vidimo da je χ_R totalna funkcija za svaku relaciju R.

Definicija

Smatramo da je relacija R izračunljiva (za neku formalizaciju izračunljivosti) ako je χ_R izračunljiva (za istu formalizaciju).
Za relacije, umjesto „izračunavanje” često kažemo „odlučivanje”.

[Prazna relacija nema jedinstvenu karakterističnu funkciju, ali nema veze jer su sve one (nulfunkcije) trivijalno izračunljive.]

Kako bismo mogli *dokazati* da su nulfunkcije izračunljive? Vrijeme je za formalizaciju ...

RAM-stroj

RAM-stroj je „matematički stroj”: idealizirano računalo s beskonačno velikom memorijom, koje nikada ne griješi.

Osnovni dijelovi RAM-stroja su:

- program ($I_0, I_1, \ldots, I_{n-1}$: fiksni konačni niz *instrukcijā*);
- registri ($\mathcal{R}_0, \mathcal{R}_1, \ldots$: beskonačno „spremnika za podatke”);
- programski brojač (PC: redni broj trenutne instrukcije).

Svaka instrukcija I_i, uz redni broj u programu ($i \in [0 \cdot\cdot n)$), ima:

- tip (INC, DEC ili GOTO);
- za INC i DEC: adresu registra $j \in \mathbb{N}$;
- za DEC i GOTO: odredište $l \in [0 \cdot\cdot n]$ (uključivo).

RAM-izračunavanje

Dakle, u RAM-programu svaka instrukcija je jednog od oblika:

INC $\mathcal{R}_j$: uvećava broj u registru („sadržaj registra”) $\mathcal{R}_j$ za 1, i ide na sljedeću instrukciju;

DEC $\mathcal{R}_j, l$: ako je sadržaj registra $\mathcal{R}_j$ pozitivan, umanjuje ga za 1 i ide na sljedeću instrukciju, a inače na instrukciju I_l;

GOTO l: ide na instrukciju I_l.

(„Ide na instrukciju I_i” zapravo znači postavljanje PC na i.)

Na početku izračunavanja, ulazni podaci se stave u registre $\mathcal{R}_1, \ldots, \mathcal{R}_k$ redom, dok se svi ostali registri, kao i PC, *resetiraju*: njihov sadržaj se postavi na 0.

Kad PC postigne vrijednost n, kažemo da izračunavanje stane, i sadržaj registra $\mathcal{R}_0$ u tom trenutku je izlazni podatak.

RAM-algoritam i relevantni registri

U samom RAM-programu nigdje ne piše intendirana mjesnost algoritma. Isti RAM-program P računa više funkcijā: po jednu za svaku mjesnost. Zato uvodimo sljedeći pojam:

RAM-algoritam P^k je uređen par RAM-programa P i mjesnosti $k \in \mathbb{N}_+$. Umjesto „P^k-izračunavanje s $\vec{x}$” kažemo samo „P-izračunavanje s $\vec{x}$” ako znamo da je duljina od $\vec{x}$ jednaka k.

Svaki algoritam P^k računa jedinstvenu funkciju: u njenoj domeni su sve k-torke s kojima P-izračunavanje stane, svaka od njih preslikana u sadržaj od $\mathcal{R}_0$ na kraju računanja.

Kako svaka instrukcija koristi najviše jedan registar, svaki RAM-program koristi samo konačno mnogo registara. Dakle za svaki RAM-program P postoji *širina* m_P: najmanji broj takav da P koristi samo registre adresā manjih od m_P.

Registar $\mathcal{R}_j$ je *relevantan* za program P ako je $j < m_P$, a relevantan je za algoritam P^k ako je $j < m_P \vee j \in [1 \cdot\cdot k]$.

Primjeri RAM-programa

Zadatak

Napišite RAM-programe koji računaju / odlučuju:

- praznu relaciju $\varnothing$
- praznu funkciju $\otimes$
- univerzalni skup $\mathbb{N}^k$
- konstantu 5 (C_5)
- identitetu I_1^1
- zbrajanje add^2
- parnost $2\mathbb{N}$
- prethodnik ($\mathrm{pd}|_{\mathbb{N}_+}$)

Odredite duljinu i širinu (n_P, m_P, m_{P^k}) napisanih programa odnosno algoritama.

Pazite!
„Odlučitelji” uvijek stanu.
„Računala” stanu točno onda kada je ulaz u domeni funkcije.

ZZA1–3

Makroi — neformalno

RAM-programi ne služe samo za računanje funkcijā: pored vrijednosti koju izračuna („stavi” u $\mathcal{R}_0$), RAM-program potencijalno proizvede i razne popratne efekte na stanjima njegovih relevantnih registara.

Gledajući te efekte, svaki RAM-program P možemo shvatiti kao elementarnu instrukciju nekog kompliciranijeg stroja.

Primjer: za svaki $j \in \mathbb{N}$, RAM-program

$$P_j := \begin{bmatrix} 0.\ \mathrm{DEC}\ \mathcal{R}_j, 2 \\ 1.\ \mathrm{GO\ TO}\ 0 \end{bmatrix}$$

možemo shvatiti kao „instrukciju” ZERO $\mathcal{R}_j$ za resetiranje registra $\mathcal{R}_j$. Kažemo da je ZERO $\mathcal{R}_j$ *makro* koji odgovara programu P_j, pišući

$$(\mathrm{ZERO}\ \mathcal{R}_j) := {P_j}^*.$$

Makro-stroj

Makro-stroj je vrlo sličan RAM-stroju: ima registre, programski brojač, *pomoćni programski brojač* AC, i makro-program.

Svaka *makro-instrukcija* (instrukcija makro-programa) je:

- ili obična RAM-instrukcija (tipa INC, DEC ili GO TO),
- ili *makro* oblika P^*, gdje je P proizvoljni RAM-program.

Makro-izračunavanje se definira kao i RAM-izračunavanje, jedino što moramo definirati ponašanje makro-stroja kad naiđe na instrukciju P^*.

Makro-stroj tada počne izvršavati program P, kao RAM-stroj s trenutnim stanjem registara i pomoćnim programskim brojačem (koji počinje od 0). Kad/ako pomoćni programski brojač dođe do n_P (odnosno kad/ako to „RAM-izračunavanje” stane), obični programski brojač makro-stroja se uveća za 1 i nastavlja se izvršavanje makro-programa od sljedeće instrukcije.

Spljoštenje (*flattening*) makro-programa

Za svaki makro-program Q, definiramo njegovo *spljoštenje* kao RAM-program $Q^\flat$ koji nastaje sljedećim postupkom:

Dok god u Q postoji barem jedan makro:

1. Maknemo prvi makro iz Q: neka je to $i.P^*$.
2. Sva odredišta veća od i i redne brojeve instrukcijā veće od i uvećamo za $n_P - 1$ (umanjimo ih za 1 ako je P prazan).
3. Za svaku instrukciju RAM-programa P, dodamo instrukciju kojoj su redni broj i odredište (ako postoji) uvećani za i.

Primijetimo da svaki prolaz kroz petlju umanjuje broj makroā u Q za 1 (jer je P RAM-program). Dakle, taj postupak sigurno stane za svaki makro-program Q.

Štoviše, Q i $Q^\flat$ su *ekvivalentni*: računaju iste funkcije.

RAM-makro ekvivalencija

Kao i u RAM-slučaju, lako možemo definirati pojmove *makro-algoritma* Q^k, i *makro-izračunljivosti* funkcije f^k.

Korolar

Funkcija je makro-izračunljiva ako i samo ako je RAM-izračunljiva.

Dokaz.

($\Leftarrow$) Neka je f^k RAM-izračunljiva funkcija, i P RAM-program koji je računa. Tada je P ujedno i makro-program (s 0 makroā), pa je f makro-izračunljiva.

($\Rightarrow$) Neka je f^k makro-izračunljiva funkcija, i Q makro-program koji je računa. Tada je $Q^\flat$ RAM-program, te iz ekvivalencije $Q^\flat$ s Q slijedi da $(Q^\flat)^k$ (skraćeno $Q^{\flat k}$) također računa funkciju f, pa je f RAM-izračunljiva. □

Primjeri makro-programa

Zadatak

Napišite makroe sa sljedećim semantikama:

- REMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ (za $i \neq j$): $r'_i = 0 \wedge r'_j = r_i$
- MMOVE n FROM $\mathcal{R}_i$.. TO $\mathcal{R}_j$.. (za $|i-j| \geq n > 0$): $(\forall t < n)(r'_{i+t} = 0 \wedge r'_{j+t} = r_{i+t})$
- CLONE $\mathcal{R}_i$ TO $\mathcal{R}_j, \mathcal{R}_k$ (za $i \neq j \neq k \neq i$): $r'_j = r'_k = r_i \wedge r'_i = 0$
- MOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ USING $\mathcal{R}_k$ (za $i \neq j \neq k \neq i$): $r'_j = r_i \wedge r'_k = 0$
- ARGS $\mathcal{R}_{j_1}, \ldots, \mathcal{R}_{j_k}$ (za $j_t > k$): $(\forall t \in [1 \cdot\cdot k])(r'_t = r_{j_t}) \wedge r'_0 = 0$

ZZ A4,6

Funkcijski makro

Definicija

Neka je $k \in \mathbb{N}_+$, f^k RAM-izračunljiva, te P_f^k RAM-algoritam koji je računa. Neka su $m, j_0, j_1, \ldots, j_k \in \mathbb{N}$. Definiramo

$$\left(P_f(\mathcal{R}_{j_1}, \ldots, \mathcal{R}_{j_k}) \to \mathcal{R}_{j_0} \text{ USING } \mathcal{R}_m..\right) := \left[\begin{array}{l} 0.\ \text{MMOVE } b \text{ FROM } \mathcal{R}_0.. \text{ TO } \mathcal{R}_b.. \\ 1.\ \text{ARGS } \mathcal{R}_{b+j_1}, \mathcal{R}_{b+j_2}, \ldots, \mathcal{R}_{b+j_k} \\ 2.\ P_f{}^* \\ 3.\ \text{REMOVE } \mathcal{R}_0 \text{ TO } \mathcal{R}_{b+j_0} \\ 4.\ \text{MMOVE } b \text{ FROM } \mathcal{R}_b.. \text{ TO } \mathcal{R}_0.. \end{array}\right]^{\flat *}$$

uz pokratu $b := \max\left\{m_{P_f^k}, m, 1 + \max_{i=0}^k j_i\right\}$.
[Širina algoritma se definira kao $m_{P^k} := \max\{m_P, 1+k\}$.]

Ekvivalencija i simulacija

Definicija

Kažemo da su (RAM- ili makro-) programi P i P′ *ekvivalentni* ako za svaki $k > 0$, algoritmi P^k i P'^k računaju istu funkciju. Kažemo da P′ *simulira* P ako za svaki $k > 0$, $\vec{x} \in \mathbb{N}^k$, te $s \in \mathbb{N}$ postoji $s' \in \mathbb{N}$ takav da je za svaki $j \in \mathbb{N}$, $r_j^{(s)} = r'^{(s')}_j$ [gdje smo sa $r_j^{(s)}$ označili sadržaj registra $\mathcal{R}_j$ nakon s koraka P-izračunavanja s $\vec{x}$ (i analogno s crticama)], te ako P-izračunavanje s $\vec{x}$ stane nakon s koraka, tada P′-izračunavanje s $\vec{x}$ stane nakon s' koraka.

Lako je, iako tehnički zahtjevno, vidjeti (indukcijom po s) da za svaki prolaz kroz petlju opisanu na prethodnom slajdu, novi Q (nakon petlje) simulira stari Q (prije petlje), te kao posljedica toga, oni su ekvivalentni. Iteracijom tog argumenta dobivamo

Teorem

Za svaki makro-program postoji ekvivalentni RAM-program.

Generalizirani makroi i makro-programi

Posljedica ekvivalencije RAM- i makro-programa je da kod pisanja makro-programa možemo koristiti ne samo makroe oblika P^* gdje je P RAM-program, već i one oblika Q^* gdje je Q neki već napisani makro-program, pri tome dakako misleći zapravo na $(Q^\flat)^*$ (skraćeno $Q^{\flat *}$).

Semantiku makro-instrukcija pišemo tako da specificiramo sadržaj registara i programskog brojača nakon izvršavanja pomoću sadržaja prije. Pritom ne pišemo registre čiji sadržaj ostaje isti, niti PC ako se samo inkrementira.

- Semantika za INC $\mathcal{R}_j$: $r'_j = r_j + 1$
- Semantika za DEC $\mathcal{R}_j$, l: ako $r_j > 0$, $r'_j = r_j - 1$; inače, $PC' = l$
- Zadatak: sami napišite semantike za GO TO l i ZERO $\mathcal{R}_j$!

Funkcijska i imperativna paradigma

Osnovni cilj prvog dijela kolegija je dokazati ekvivalentnost dvaju (na prvi pogled jako različitih) poimanja izračunljivosti:

- aksiomatskog: „funkcija sastavljena od očito izračunljivih funkcijā pomoću pravilā koja očito čuvaju izračunljivost"
- operativnog: „funkcija koju računa neki algoritam sastavljen od jednostavnih mehaničkih instrukcija"

Zapravo želimo dokazati ekvivalentnost dviju paradigmi: funkcijskog i imperativnog programiranja.

- Za smjer „funkcijsko simulira imperativno", osnovna ideja je simulacija protoka vremena: shvatimo redni broj koraka kao još jedan ulazni podatak. [To ćemo napraviti kasnije.]
- Za smjer „imperativno simulira funkcijsko", osnovna ideja je simulacija stoga poziva (*call stack*), i funkcijski makro.

Semantika funkcijskog makroa

Propozicija

Semantika funkcijskog makroa, uz oznake s prethodnog slajda, te pokratu $\vec{r} := (r_{j_1}, r_{j_2}, \ldots, r_{j_k})$, jest:

1. Ako je $\vec{r} \in \mathcal{D}_f$, tada je $r'_{j_0} = f(\vec{r}) \wedge (\forall t \in [b \cdot\cdot 2b))(r'_t = 0)$. [Zbog $b > m$, za sve $i \in [0 \cdot\cdot m) \setminus \{j_0\}$ vrijedi $r'_i = r_i$.]
2. Ako $\vec{r} \notin \mathcal{D}_f$, izvršavanje funkcijskog makroa ne stane.

	r_0	$r_1, \ldots, r_k$	r_{j_0}	r_b	r_{b+j_0}	r_{2b-1}
0. MMOVE	0	$0, \ldots, 0$	0	r_0	r_{j_0}	r_{b-1}
1. ARGS	0	$r_{j_1}, \ldots, r_{j_k}$	0	r_0	r_{j_0}	r_{b-1}
2. P_f^*	$f(\vec{r})$	$?, \ldots, ?$	?	r_0	r_{j_0}	r_{b-1}
3. REMOVE	0	$?, \ldots, ?$	?	r_0	$f(\vec{r})$	r_{b-1}
4. MMOVE	r_0	$r_1, \ldots, r_k$	$f(\vec{r})$	0	0	0

Inicijalne funkcije

Kao što rekosmo, izračunljivost možemo definirati i aksiomatski. *Inicijalne* funkcije čiju izračunljivost podrazumijevamo su:

- nulfunkcija $Z^1 = \chi_{\varnothing^1}$, zadana sa $Z(x) := 0$;
- sljedbenik Sc^1, zadana sa $Sc(x) := x + 1$;
- za $1 \le n \le k$, koordinatna projekcija I^k_n, zadana s $I_n(x_1, x_2, \ldots, x_k) := x_n$ (reprezentacija $id_{\mathbb{N}^+}$).

Sve inicijalne funkcije su totalne: $\mathcal{D}_Z = \mathcal{D}_{Sc} = \mathbb{N}$, a $\mathcal{D}_{I^k_n} = \mathbb{N}^k$.

Propozicija

Svaka inicijalna funkcija je RAM-izračunljiva.

Kompozicija — intuitivno

Htjeli bismo reći: kompozicija dviju izračunljivih funkcija, G^k i H^l, je izračunljiva funkcija. Ali postoje dvije komplikacije.

Prvo, da bi kompozicija bila definirana, kodomena funkcije G morala bi biti $\mathbb{N}^l$. U našem formalizmu to znači da je zadajemo pomoću l funkcijā iste mjesnosti: $G^k_1, \ldots, G^k_l$.

Drugo, moramo paziti na domene: G_i i H nisu nužno totalne. Kompozicija se formalno definira pomoću ulančanih uređenih parova [*marljiva* evaluacija], ne „pravilima preslikavanja".

Primjer

Recimo, kompozicija $I^2_1 \circ (Z, \otimes^1)$ nije definirana nigdje — bez obzira na to što $I_1(x, y) = x$ ne koristi svoj drugi argument.

Kompozicija — formalno

Definicija

Neka su $k, l \in \mathbb{N}_+$, te $G^k_1, \ldots, G^k_l$ i H^l funkcije. Za funkciju F^k:

$$\mathcal{D}_F := \left\{ \vec{x} \in \bigcap_{i=1}^{l} \mathcal{D}_{G_i} \;\middle|\; \big(G_1(\vec{x}), \ldots, G_l(\vec{x})\big) \in \mathcal{D}_H \right\}$$
$$F(\vec{x}) := H\big(G_1(\vec{x}), \ldots, G_l(\vec{x})\big) \text{ za sve } \vec{x} \in \mathcal{D}_F$$

kažemo da je dobivena *kompozicijom* funkcija $G_1, \ldots, G_l$ i H. Pišemo $F := H \circ (G_1, \ldots, G_l)$.

Kažemo da je skup funkcijā $\mathcal{F}$ *zatvoren na kompoziciju* ako za sve $k, l \in \mathbb{N}_+$, za sve k-mjesne $G_1, \ldots, G_l \in \mathcal{F}$, za sve l-mjesne $H \in \mathcal{F}$, vrijedi $H \circ (G_1, \ldots, G_l) \in \mathcal{F}$.

Iz definicije $\mathcal{D}_F$: kompozicija totalnih funkcija je totalna (skup totalnih funkcija zatvoren je na kompoziciju).

Parcijalna jednakost

Definicija s prethodnog slajda djeluje nezgrapno. Prvo definiramo domenu D, a zatim na D definiramo funkciju izrazom koji jedino ima vrijednost na D.

Možemo pisati samo izraz, podrazumijevajući da lijeva strana ima vrijednost ako i samo ako je ima i desna, te su tada te vrijednosti jednake. Da bismo naglasili tu interpretaciju, pišemo $\simeq$ umjesto $=$ u „točkovnoj" definiciji:

$$\big(H \circ (G_1, \ldots, G_l)\big)(\vec{x}) \simeq H\big(G_1(\vec{x}), \ldots, G_l(\vec{x})\big)$$

Precizno, „izraz ima vrijednost" znači da za svaki njegov podizraz oblika $f(\vec{v})$ (možda zapisan, npr. infiksno kao $v_1 + v_2$) vrijedi da (rekurzivno) svaki izraz v_i ima vrijednost, te $\vec{v} \in \mathcal{D}_f$.

Ovo pravilo se *ne odnosi* na izraze definirane po slučajevima, već samo na pojedine slučajeve — više o tome reći ćemo kasnije.

RAM-izračunljivost kompozicije

Propozicija

Skup RAM-izračunljivih funkcija je zatvoren na kompoziciju.

Dokaz.

Neka su $k, l \in \mathbb{N}_+$, te neka su $G^k_1, \ldots, G^k_l$ i H^l RAM-izračunljive. Dakle, postoje RAM-algoritmi $P^k_{G_1}, \ldots, P^k_{G_l}$ i P^l_H koji ih računaju. Tada RAM-algoritam

$$\left[\begin{array}{r} 0.\ P_{G_1}(\mathcal{R}_1, \ldots, \mathcal{R}_k) \to \mathcal{R}_{k+1} \text{ USING } \mathcal{R}_{k+l+1}.. \\ 1.\ P_{G_2}(\mathcal{R}_1, \ldots, \mathcal{R}_k) \to \mathcal{R}_{k+2} \text{ USING } \mathcal{R}_{k+l+1}.. \\ \vdots \\ (l-1).\ P_{G_l}(\mathcal{R}_1, \ldots, \mathcal{R}_k) \to \mathcal{R}_{k+l} \text{ USING } \mathcal{R}_{k+l+1}.. \\ l.\ P_H(\mathcal{R}_{k+1}, \ldots, \mathcal{R}_{k+l}) \to \mathcal{R}_0 \text{ USING } \mathcal{R}_{k+l+1}.. \end{array}\right]^{\flat k}$$

računa $H \circ (G_1, \ldots, G_l)$, koja je zato RAM-izračunljiva. □

Primitivna rekurzija

Definicija

Neka je $k \in \mathbb{N}_+$, te G^k i H^{k+2} totalne funkcije. Za (jedinstvenu) funkciju F^{k+1} definiranu jednakostima

$$F(\vec{x}, 0) := G(\vec{x})$$
$$F(\vec{x}, y + 1) := H\big(\vec{x}, y, F(\vec{x}, y)\big)$$

kažemo da je dobivena *primitivnom rekurzijom* iz funkcijā G i H. Pišemo $F := G \,\mathrm{PR}\, H$. (PR je manjeg prioriteta od $\circ$.)

Kažemo da je skup funkcijā $\mathcal{F}$ *zatvoren na primitivnu rekurziju* ako za sve $k \in \mathbb{N}_+$, za sve totalne k-mjesne $G \in \mathcal{F}$, za sve totalne $(k+2)$-mjesne $H \in \mathcal{F}$, vrijedi $G \,\mathrm{PR}\, H \in \mathcal{F}$.

Indukcijom po y se može dokazati da je $G \,\mathrm{PR}\, H$ uvijek totalna.

Primitivnom rekurzijom zasad ne možemo definirati jednomjesne funkcije (jer je $k + 1 > 1$). To ćemo riješiti poslije.

Točkovne defincije funkcija

Često je lakše zapisati definiciju funkcije kad imamo imena za argumente. Važno je imati na umu da se svaka takva „točkovna" definicija funkcije uvijek mora moći zapisati *simbolički*, eliminirajući sva korištenja imena argumenata.

Točkovna definicija je često čitljivija, ali postoji opasnost da nećemo vidjeti da je neki argument nemoguće eliminirati. Recimo ako imamo funkcije $G_0, G_1, G_2, \ldots$, tada $F(x) \simeq G_x(x)$ nije dobra točkovna definicija od F, jer ne možemo eliminirati x. [I doista, ako su sve G_i izračunljive, F ne mora biti izračunljiva!] Takva opasnost ne postoji kod simboličke definicije.

Primjer

$$\left.\begin{array}{r} f(x, y, 0, z) := g\big(x, h(x, y, z)\big) \\ f(x, y, t + 1, z) := h\big(z, f(x, y, t, z), g(t, z)\big) \end{array}\right\} \leftrightsquigarrow$$
$$f^4 := \big(g^2 \circ (I^3_1, h^3) \,\mathrm{PR}\, h^3 \circ \big(I^5_3, I^5_5, g^2 \circ (I^5_4, I^5_3)\big)\big) \circ (I^4_1, I^4_2, I^4_4, I^4_3)$$

Primitivno rekurzivne funkcije

Definicija

Najmanji skup funkcijā koji sadrži sve inicijalne funkcije, te je zatvoren na kompoziciju i primitivnu rekurziju, zovemo skupom *primitivno rekurzivnih* funkcija.

Propozicija

Funkcija je primitivno rekurzivna ako i samo ako se može dobiti iz inicijalnih funkcija pomoću konačno mnogo primjena kompozicije i primitivne rekurzije (ima *simboličku definiciju*).

Tako ćemo uvijek dokazivati primitivnu rekurzivnost funkcijā koje proučavamo — jedino što nećemo uvijek sve iznova raspisivati do inicijalnih funkcijā, nego ćemo koristiti funkcije čiju smo primitivnu rekurzivnost već utvrdili.

Aritmetičke operacije su primitivno rekurzivne

Zadatak

Napišite simboličke i točkovne definicije za:

- konstantne funkcije, $C_n^k(\vec{x}) := n$ (za $n \in \mathbb{N}$, $k \in \mathbb{N}_+$)
- zbrajanje, $\mathrm{add}(x,y) := x + y$ [Odgovor: $\mathrm{add}^2 = I_1^1 \overset{PR}{\circ} Sc \circ I_3^3$]
- množenje, $\mathrm{mul}(x,y) := x \cdot y$
- potenciranje, $\mathrm{pow}(x,y) := x^y$

Mnogi skupovi funkcija imaju svojstvo da sadrže sve inicijalne funkcije, i zatvoreni su na kompoziciju i primitivnu rekurziju. Čim to možemo zaključiti za neki skup funkcija $\mathcal{F}$, znamo da je skup primitivno rekurzivnih funkcija podskup od $\mathcal{F}$. Recimo:

Propozicija

1. Sve primitivno rekurzivne funkcije su totalne.
2. Sve primitivno rekurzivne funkcije su RAM-izračunljive.

Još neke primitivno rekurzivne funkcije i relacije

Zadatak

Napišite simboličke odnosno točkovne definicije za sljedeće funkcije / karakteristične funkcije sljedećih relacija:

- prethodnik, $\mathrm{pd}(x) := \max\{x-1, 0\}$
- pozitivnost $\mathbb{N}_+$
- faktorijel, $\mathrm{factorial}(x) := x!$
- ograničeno oduzimanje, $x \dot{-} y := \mathrm{sub}(x,y) := \max\{x-y, 0\}$
- obrnuti strogi uređaj $>$
- strogi uređaj $<$

ZZB1

Parcijalno rekurzivne funkcije

Definicija

Najmanji skup funkcija koji sadrži sve inicijalne funkcije, te je zatvoren na kompoziciju, primitivnu rekurziju i minimizaciju, naziva se skup *parcijalno rekurzivnih* funkcija.

Propozicija (Simbolička definicija)

Funkcija je parcijalno rekurzivna ako je korijen *stabla* čiji listovi su inicijalne funkcije, a svaki čvor je funkcija dobivena

- kompozicijom iz svoje djece (2 ili više funkcija)
- primitivnom rekurzijom iz svoje djece (2 totalne funkcije)
- minimizacijom svog djeteta (1 relacija)

ili relacija čija je karakteristična funkcija dobivena na isti način.

Imperativno programiranje simulira funkcijsko

Teorem

Svaka parcijalno rekurzivna funkcija je RAM-izračunljiva.

Plan dokaza.

Znamo da skup RAM-izračunljivih funkcija sadrži sve inicijalne funkcije i da je zatvoren na kompoziciju i primitivnu rekurziju. Još treba dokazati zatvorenost na minimizaciju: učinite to! □

Postorder-obilaskom simboličke definicije parcijalno rekurzivne funkcije možemo je *kompilirati* u makro-program, pa onda i spljoštiti (*linkati*) u RAM-program.

Kao što umjesto raspisivanja uvijek do inicijalnih funkcija možemo koristiti već dokazano rekurzivne funkcije, tako možemo i neke često korištene *operatore* (preslikavanja funkcija/relacija) zapisati pomoću osnovna tri. Krenimo!

Degenerirana primitivna rekurzija

Propozicija

Neka je $a \in \mathbb{N}$, te H^2 primitivno rekurzivna funkcija. Tada je funkcija F^1, definirana jednakostima

$$F(0) := a$$
$$F(x+1) := H\big(x, F(x)\big)\,,$$

također primitivno rekurzivna. (Pišemo $F := a \overset{PR}{\circ} H$.)

Dokaz.

$$F = \big(C_a^1 \overset{PR}{\circ} \overbrace{H \circ (I_2^3, I_3^3)}^{\text{primitivno rekurzivna, dakle totalna}}\big) \circ (Z, I_1^1)\,.$$

Ideja je uvesti „irelevantni" parametar na početak, koji ne radi ništa osim što uvećava sve mjesnosti za 1. □

Minimizacija relacije

Sve dosad definirane operacije su čuvale totalnost. Minimizacija pruža mogućnost dobivanja parcijalnih funkcija.

Definicija

Neka je $k \in \mathbb{N}_+$ i R^{k+1} relacija. Za funkciju f^k definiranu s

$$\mathcal{D}_f := \exists_* R := \{\vec{x} \in \mathbb{N}^k \mid \exists y\, R(\vec{x}, y)\} \quad (\textit{projekcija} \text{ relacije } R)$$
$$f(\vec{x}) := \min\{y \in \mathbb{N} \mid R(\vec{x}, y)\}, \text{ za sve } \vec{x} \in \exists_* R$$

kažemo da je dobivena *minimizacijom* relacije R.
Pišemo $f := \mu R$, točkovno $f(\vec{x}) \simeq \mu y\, R(\vec{x}, y)$.

Kažemo da je skup funkcijā $\mathcal{F}$ *zatvoren na minimizaciju* ako za sve $k \in \mathbb{N}_+$, za sve relacije R^{k+1}, $\chi_R \in \mathcal{F}$ povlači $\mu R \in \mathcal{F}$.

Zbog dobre uređenosti od $\mathbb{N}$, projekcija $\exists_* R$ je točno skup svih $\vec{x}$ za koje izraz $\mu y R(\vec{x}, y)$ ima vrijednost.

Rekurzivne funkcije i relacije

Definicija

Parcijalno rekurzivne funkcije koje su totalne zovemo *rekurzivnima*. Relacije čije karakteristične funkcije su (parcijalno) rekurzivne također zovemo rekurzivnima.

Pojam „parcijalno rekurzivna relacija" je beskoristan: sve takve relacije su rekurzivne (jer je χ_R uvijek totalna).

Zadatak

Dokažite: svaka primitivno rekurzivna funkcija je rekurzivna.

Zadatak

Grafički prikažite odnos svih skupova brojevnih funkcija koje smo dosad uveli.

Negacija / komplement

Propozicija (2 komada)

Neka je $k \in \mathbb{N}_+$, te R^k (primitivno) rekurzivna relacija. Tada je i relacija $(R^c)^k$ zadana s $R^c(\vec{x}) :\Leftrightarrow \neg R(\vec{x})$ (primitivno) rekurzivna.

[$(\varnothing^k)^c$ je definirana, kao $\mathbb{N}^k$, samo za fiksnu mjesnost k!]

Fraza „(primitivno)" znači skraćeni zapis za dvije propozicije: jedna za primitivno rekurzivne, a druga za rekurzivne relacije.

Dokaz.

Funkcija zadana s $f_0(x) := 1 \dot{-} x$ je primitivno rekurzivna ($f_0 = \mathrm{sub} \circ (C_1^1, I_1^1)$), i vrijedi $\chi_{R^c} = f_0 \circ \chi_R$.

Dakle, ako je χ_R primitivno rekurzivna, takva je i χ_{R^c}. Ako je pak χ_R rekurzivna, takva je i χ_{R^c}. □

Ostale logičke / skupovne operacije

Propozicija

Neka je $k \in \mathbb{N}_+$, te neka su R^k i P^k (primitivno) rekurzivne relacije. Tada su i sljedeće relacije (primitivno) rekurzivne:

$$Q_1(\vec{x}) :\Longleftrightarrow R(\vec{x}) \wedge P(\vec{x}) \qquad Q_1 := R \cap P$$
$$Q_2(\vec{x}) :\Longleftrightarrow R(\vec{x}) \vee P(\vec{x}) \qquad Q_2 := R \cup P$$
$$Q_3(\vec{x}) :\Longleftrightarrow R(\vec{x}) \to P(\vec{x}) \qquad Q_5 := R \setminus P$$
$$Q_4(\vec{x}) :\Longleftrightarrow R(\vec{x}) \leftrightarrow P(\vec{x}) \qquad Q_6 := R \triangle P$$

Zadatak (skica dokaza).

Imitirajte prethodni dokaz: za svaki $i \in [1 \mathinner{..} 6]$ nađite primitivno rekurzivnu funkciju f_i^2 takvu da je $\chi_{Q_i} = f_i \circ (\chi_R, \chi_P)$. □

Višestruke skupovne operacije

Lema

Za svaki $k \in \mathbb{N}_+$, funkcije add^k i mul^k (zbrajanje i množenje k brojeva) primitivno su rekurzivne.

Dokaz.

Indukcijom po k. Naravno, $\mathsf{mul}^1 = \mathsf{I}_1^1$. Za $k \geq 2$,

$$\mathsf{mul}(x_1, \ldots, x_{k+1}) := \mathsf{mul}\big(\mathsf{mul}(x_1, \ldots, x_k), x_{k+1}\big),$$

dakle mul^{k+1} je dobivena kompozicijom iz mul^2, mul^k i koordinatnih projekcija. Za add^k, potpuno analogno. □

Sada se istom tehnikom može dokazati

Propozicija

Neka su $k, l \in \mathbb{N}_+$, te neka su $R_1^k, \ldots, R_l^k$ (primitivno) rekurzivne relacije. Tada su i $\bigcap_{i=1}^l R_i$ i $\bigcup_{i=1}^l R_i$ (primitivno) rekurzivne.

Grananje („definicija po slučajevima")

Definicija

Neka su $k, l \in \mathbb{N}_+$, te neka su $G_0^k, \ldots, G_l^k$ funkcije, i $R_1^k, \ldots, R_l^k$ u parovima disjunktne relacije. Za funkciju definiranu s

$$R_0 := \Big(\bigcup_{i=1}^l R_i\Big)^c \qquad F(\vec{x}) := \begin{cases} G_1(\vec{x}), & R_1(\vec{x}) \\ \quad\vdots & \\ G_l(\vec{x}), & R_l(\vec{x}) \\ G_0(\vec{x}), & \text{inače} \end{cases} \quad \text{za sve } \vec{x} \in \mathcal{D}_F$$
$$\mathcal{D}_F := \bigcup_{i=0}^l \big(\mathcal{D}_{G_i} \cap R_i\big)$$

kažemo da je definirana *grananjem* iz *granā* G_i i *uvjetā* R_i.

Pišemo simbolički

$$F := \mathsf{if}\big\{R_1 : G_1, \ldots, R_l : G_l, G_0\big\}.$$

Ako ne navedemo G_0, podrazumijevamo $G_0 := \otimes^k$.

Domena funkcije definirane grananjem

Prethodnu definiciju također možemo zapisati pomoću parcijalne jednakosti $:\simeq$, ali moramo biti oprezni pri njenom razumijevanju. Desna strana ne mora imati svojstvo da *svi* njezini podizrazi trebaju imati vrijednost da bi i lijeva strana imala vrijednost — samo „relevantni redak" (onaj koji odgovara relaciji koja vrijedi za $\vec{x}$) treba imati to svojstvo.

Ipak, kao i kod kompozicije, komplikacije s domenom bit će bitne kasnije. Zasad radimo s (primitivno) rekurzivnim funkcijama, koje su totalne — a lako je vidjeti da je funkcija dobivena grananjem iz totalnih funkcija uvijek totalna. (Samo tada ne smijemo ispustiti G_0, jer $\otimes^k$ *nije* totalna!)

Teorem o grananju, (primitivno) rekurzivna verzija

Teorem

Neka su $k, l \in \mathbb{N}_+$,
neka su $G_0^k, G_1^k, \ldots, G_l^k$ (primitivno) rekurzivne funkcije,
te neka su $R_1^k, \ldots, R_l^k$ u parovima disjunktne (primitivno) rekurzivne relacije.
Tada je funkcija $F^k := \mathsf{if}\big\{R_1 : G_1, \ldots, R_l : G_l, G_0\big\}$ također (primitivno) rekurzivna.

Dokaz.

Prvo primijetimo da je $R_0 := \big(\bigcup_{i=1}^l R_i\big)^c$ (primitivno) rekurzivna kao komplement unije l (primitivno) rekurzivnih relacija. Tada

$$F = G_0 \cdot \chi_{R_0} + G_1 \cdot \chi_{R_1} + \cdots + G_l \cdot \chi_{R_l},$$

odnosno F je dobivena kompozicijom add^{l+1}, mul^2, G_i i χ_{R_i}. □

Konačne relacije

Zadatak

Dokažite da su sljedeće relacije primitivno rekurzivne:

- nestrogi uređaj $\leq$
- obrnuti nestrogi uređaj $\geq$
- jednakost $=$
- jednočlana jednomjesna relacija $\{c\}$ (za $c \in \mathbb{N}$)
- jednočlana relacija $\{\vec{c}\}$ (za $\vec{c} \in \mathbb{N}^k$, $k \in \mathbb{N}_+$)
- prazna relacija $\varnothing^k$ (za $k \in \mathbb{N}_+$)
- konačna relacija $\{\vec{c}_1, \ldots, \vec{c}_l\} \subseteq \mathbb{N}^k$ (za $k \in \mathbb{N}_+$, $l \in \mathbb{N}$)

ZZB2

Editiranje (konačna promjena) totalnih funkcija

Propozicija

Neka je $k \in \mathbb{N}_+$, neka je G^k (primitivno) rekurzivna funkcija, te F^k totalna funkcija koja se podudara s G u svima osim konačno mnogo točaka. Tada je i F (primitivno) rekurzivna.

Dokaz.

Po pretpostavci, skup $\{\vec{x}^k \mid F(\vec{x}) \neq G(\vec{x})\}$ je konačan: označimo mu sve elemente (npr. leksikografskim redom) s $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_l$. Ako je taj skup prazan, $F = G$ pa nemamo što dokazivati. Inače,

$$F = \mathsf{if}\big\{\{\vec{c}_1\} : C^k_{F(\vec{c}_1)}, \ldots, \{\vec{c}_l\} : C^k_{F(\vec{c}_l)}, G\big\},$$

(primitivno) rekurzivna funkcija po teoremu o grananju. □

Zadatak: Gdje se u dokazu koristi totalnost funkcije F?

Ograničene sume, produkti, brojenje, minimizacija, ...

Propozicija

Neka je $k \in \mathbb{N}_+$ te G^k (primitivno) rekurzivna funkcija, odnosno R^k (primitivno) rekurzivna relacija.
Tada su (primitivno) rekurzivne i sljedeće funkcije/relacije:

$$F_\Sigma^k(\vec{x}, y) := \textstyle\sum_{i<y} G(\vec{x}, i) \quad \text{i} \quad F_\Pi^k(\vec{x}, y) := \prod_{i<y} G(\vec{x}, i),$$
$$F_\mu^k(\vec{x}, y) := \mu i \big(i < y \to R(\vec{x}, i)\big) =: (\mu i < y)\, R(\vec{x}, i),$$
$$F_\#^k(\vec{x}, y) := \operatorname{card}\{i \in [0 \mathinner{..} y) \mid R(\vec{x}, i)\} =: (\# i < y)\, R(\vec{x}, i),$$
$$Q_\exists^k(\vec{x}, y) :\Leftrightarrow (\exists i < y)\, R(\vec{x}, i) \quad \text{i} \quad Q_\forall^k(\vec{x}, y) :\Leftrightarrow (\forall i < y)\, R(\vec{x}, i).$$

Jedna primitivna rekurzija:

$$F_\mu(\vec{x}, 0) = 0\,, \quad F_\mu(\vec{x}, y+1) = \begin{cases} F_\mu(\vec{x}, y), & R\big(\vec{x}, F_\mu(\vec{x}, y)\big) \\ y+1, & \text{inače} \end{cases}.$$

(Primitivno) rekurzivne granice

Napomena

Kompozicijom s nekom (primitivno) rekurzivnom funkcijom H na mjestu argumenta y lako dobijemo (primitivnu) rekurzivnost zbrajanja / množenja / brojenja / minimizacije / kvantifikacije do (primitivno) rekurzivne granice. Recimo,

$$\big(\#\, i < H(\vec{x})\big)\, R(\vec{x}, y) = F_{\#}\big(\vec{x}, H(\vec{x})\big)\,.$$

Pišući $Sc \circ H$ za granicu umjesto H, dobijemo sve te rezultate i za uključene gornje granice, jer je $y \le H(\vec{x}) \iff y < Sc\big(H(\vec{x})\big)$.

Napomena

Neograničena kvantifikacija $\exists_* R$ (čak ni primitivno) rekurzivne relacije R ne mora biti rekurzivna! To ćemo dokazati kasnije.

ZZB3,4

Kodiranje — općenito

Neka je $\mathcal{K}$ skup koji ne sadrži prirodne brojeve nego neke druge objekte: racionalne brojeve, RAM-programe, regularne izraze nad fiksnom abecedom, konačne nizove prirodnih brojeva, ...

Kodiranje skupa $\mathcal{K}$ je izračunljiva injekcija $\mathbb{N}\mathcal{K} := \kappa : \mathcal{K} \to \mathbb{N}$, kojoj je slika („skup svih kodova") $\mathcal{I}_\kappa$ rekurzivan skup, a parcijalni inverz $\kappa^{-1} : \mathcal{I}_\kappa \to \mathcal{K}$ je također izračunljiv.

[Očito, da bi kodiranje postojalo, skup $\mathcal{K}$ mora biti prebrojiv!]

Ova definicija nije stroga, jer govori o postojanju intuitivnih algoritama, čiji ulazi/izlazi nisu prirodni brojevi. $\mathcal{K}$ je najčešće neki apstraktni tip podataka: bitno je njegove osnovne metode prikazati kao rekurzivne funkcije na kodovima. ZZC4

Kodiranje konačnih nizova prirodnih brojeva

Kao važan primjer uzmimo skup $\mathbb{N}^* := \bigcup_{i \in \mathbb{N}} \mathbb{N}^i$.

Njegovo kodiranje predstavimo familijom funkcijā $Code^k, k \in \mathbb{N}_+$ (i konstantom $\langle\rangle := 1$ za kod praznog niza), a za osnovne metode uzmimo duljinu (lh) i indeksiranje ($part$).

S p_i označimo i-ti prim-broj ($p_0 = 2$). Za $\vec{x} \in \mathbb{N}^k$, označimo

$$Code^k(\vec{x}) := \langle x_1, \ldots, x_k \rangle := 2^{x_1+1} \cdot 3^{x_2+1} \cdot 5^{x_3+1} \cdots p_{k-1}^{x_k+1}\,.$$

Propozicija

Za svaki $k \in \mathbb{N}_+$, funkcija $Code^k$ je primitivno rekurzivna.

Dekodiranje se svodi na rastav na prim-faktore (prethodnici eksponenata su članovi niza). [Sljedbenici u eksponentima služe za injektivnost: bez njih bi $(1,2)$ i $(1,2,0)$ imali isti kod 18.]

Povijest funkcije

Često ne želimo kodiranje fiksne mjesnosti, nego ono čiji su argumenti zadani nekom funkcijom.

Definicija

Neka je $k \in \mathbb{N}_+$ i G^k totalna funkcija. Za funkciju F^k zadanu s

$$F(\vec{x}, y) := \langle G(\vec{x}, 0), G(\vec{x}, 1), \ldots, G(\vec{x}, y-1) \rangle$$

kažemo da je *povijest* funkcije G i pišemo $F := \overline{G}$.

Zadatak

Izračunajte $\overline{add}(2,1,3)$, $\overline{Z}(5)$, $\overline{Code}(1,1)$, $\overline{mul}(9,7,0)$, $\overline{\overline{pow}}(8,2)$.

Za dokaz da operator povijesti čuva (primitivnu) rekurzivnost, ne možemo koristiti $Code^y$ jer mjesnost mora biti fiksna — prvo nam treba nekoliko pojmova iz teorije brojeva.

Dijeljenje s ostatkom, prim-brojevi

Zadatak

Dokažite da su sljedeće funkcije/operacije i relacije primitivno rekurzivne:

- zamjena nule jedinicom, $x^{\bullet} := \max\{x, 1\}$
- cjelobrojno dijeljenje, $x \mathbin{/\!/} y := \left\lfloor \frac{x}{y^{\bullet}} \right\rfloor$
- ostatak cjelobrojnog dijeljenja, $x \bmod y$
- djeljivost, $y \mid x$
- skup prim-brojeva $\mathbb{P}$
- $nextprime(p) :=$ prvi sljedeći prim-broj nakon p
- $prime(n) := p_n$
- $ex(n,i) :=$ eksponent od p_i u rastavu $n^{\bullet}$ na prim-faktore

Jedna točkovna definicija: $x \mathbin{/\!/} y = pd\big((\mu t \le x)(t \cdot y > x)\big)$.

Osnovne metode konačnih nizova: lh i part

Za $c = \langle x_1, x_2, \ldots, x_k \rangle \in Seq := \mathcal{I}_{\langle \ldots \rangle}$, označimo $lh(c) := k$ i

$c[i] := part(c,i) := \begin{cases} x_{i+1}, & i < k \\ 0, & \text{inače} \end{cases}$. To je *parcijalna specifikacija*, jer nismo rekli ništa o tome kako lh i $part$ djeluju za $c \notin Seq$ — ali *postoje* primitivno rekurzivne funkcije koje to zadovoljavaju. Nađite ih! Ubuduće ćemo često parcijalno specificirati funkcije.

Lema (o povijesti)

Ako je G (primitivno) rekurzivna funkcija, takva je i $\overline{G}$.

Korolar

Skup Seq svih kodova konačnih nizova primitivno je rekurzivan.

Konkatenacija

Propozicija

Postoji primitivno rekurzivna operacija $*$ takva da za sve $\vec{x}, \vec{y} \in \mathbb{N}^*$ vrijedi $\langle \vec{x} \rangle * \langle \vec{y} \rangle = \langle \vec{x}, \vec{y} \rangle$.

[Operacija $*$ je također parcijalno specificirana, na $Seq \times Seq$.]

Dokaz.

Definiramo funkciju G^4 grananjem

$$G(x, y, k, i) := \begin{cases} x[i], & i < k \\ y[i \dot{-} k], & \text{inače} \end{cases}.$$

Sada možemo definirati $x * y := \overline{G}\big(x, y, lh(x), lh(x) + lh(y)\big)$.

Po teoremu o grananju G je primitivno rekurzivna, pa je $\overline{G}$ također takva po lemi o povijesti, a onda i $*$. □

Rekurzija s poviješću

Propozicija

Neka je $k \in \mathbb{N}_+$, te G^k (primitivno) rekurzivna funkcija. Tada je (primitivno) rekurzivna i funkcija F^k, zadana jednadžbom

$$F(\vec{x}, y) := G\big(\vec{x}, \overline{F}(\vec{x}, y)\big).$$

Dokaz.

Povijest funkcije F zadovoljava primitivnu rekurziju

$$\overline{F}(\vec{x}, 0) = \langle\rangle = 1$$
$$\overline{F}(\vec{x}, y+1) = \overline{F}(\vec{x}, y) * \big\langle G\big(\vec{x}, \overline{F}(\vec{x}, y)\big)\big\rangle$$

pa je (primitivno) rekurzivna jer su C_1, G, $*$ i $Code^1$ takve. Tad je i F takva kao kompozicija G, $\overline{F}$ i koordinatnih projekcija. □

Simultana primitivna rekurzija

Propozicija

Neka su $k, l \in \mathbb{N}_+$, te neka su $G_1^k, \ldots, G_l^k$ i $H_1^{k+l+1}, \ldots, H_l^{k+l+1}$ (primitivno) rekurzivne funkcije. Tada su (primitivno) rekurzivne i funkcije $F_1^{k+1}, \ldots, F_l^{k+1}$, zadane jednadžbama

$$F_i(\vec{x}, 0) := G_i(\vec{x}),$$
$$F_i(\vec{x}, y+1) := H_i\big(\vec{x}, y, F_1(\vec{x}, y), \ldots, F_l(\vec{x}, y)\big).$$

Dokaz.

Funkcija $G^k := \mathsf{Code}^l \circ (G_1, \ldots, G_l)$, i funkcija H^{k+2}, zadana s

$$H(\vec{x}, y, z) := \big(\mathsf{Code}^l \circ (H_1, \ldots, H_l)\big)(\vec{x}, y, z[0], \ldots, z[l-1]),$$

su (primitivno) rekurzivne. Tada je $F_i = \mathsf{part} \circ (G \mathbin{\mathbb{R}} H, C_{i-1}^{k+1})$. □

Kodiranje RAM-programa

Programi su konačni nizovi instrukcija, pa kompozicijom kodiranja instrukcija i konačnih nizova dobivamo kodiranje RAM-programa.

Definicija

Za $P = \big[\ t.\ I_t\ \big]_{t<n}$, definiramo $\ulcorner P \urcorner := \langle \ulcorner I_0 \urcorner, \ldots, \ulcorner I_{n-1} \urcorner \rangle$.

Propozicija

$\ulcorner \cdots \urcorner$ je injekcija s primitivno rekurzivnom slikom.

Skica dokaza (uputa):

$$\mathsf{Prog}(e) :\Leftrightarrow \mathsf{Seq}(e) \wedge \big(\forall i < \mathsf{lh}(e)\big)\big(\mathsf{Ins}(e[i]) \wedge \mathsf{dest}(e[i]) \le \mathsf{lh}(e)\big)$$

Kodiranje instrukcija RAM-stroja

Definicija

- $\ulcorner \textsc{inc}\ \mathcal{R}_j \urcorner := \mathsf{codeINC}(j) := \langle 0, j \rangle = 6 \cdot 3^j$
- $\ulcorner \textsc{dec}\ \mathcal{R}_j, l \urcorner := \mathsf{codeDEC}(j, l) := \langle 1, j, l \rangle = 60 \cdot 3^j \cdot 5^l$
- $\ulcorner \textsc{go to}\ l \urcorner := \mathsf{codeGOTO}(l) := \langle 2, l \rangle = 24 \cdot 3^l$

Zadatak

Dokažite primitivnu rekurzivnost sljedećih skupova i funkcija:

- codeINC, codeDEC, codeGOTO
- InsINC (skup kodova instrukcija tipa INC)
- InsDEC i InsGOTO (analogno, za tipove DEC i GO TO)
- Ins (skup *svih* kodova instrukcija)
- $\mathsf{regn}(\ulcorner \textsc{inc}\ \mathcal{R}_j \urcorner) = \mathsf{regn}(\ulcorner \textsc{dec}\ \mathcal{R}_j, l \urcorner) = j$, inače 0
- $\mathsf{dest}(\ulcorner \textsc{dec}\ \mathcal{R}_j, l \urcorner) = \mathsf{dest}(\ulcorner \textsc{go to}\ l \urcorner) = l$, inače 0

Kodiranje stanja registara

Sada za $\mathcal{K}$ uzmimo skup nizova prirodnih brojeva *s konačnim nosačem* (takvih da su nula od nekog mjesta nadalje).
Taj skup kodiramo kao $\mathbb{N}^*$, samo bez sljedbenika u eksponentu:

$$\langle x_0, x_1, x_2, \ldots, 0, 0, 0, \ldots \rangle := \prod_{i \in \mathbb{N}} p_i^{x_i}.$$

Za svaki RAM-stroj $\mathcal{S}$, u svakom koraku rada, stanje registara je s konačnim nosačem (najviše m_{P^k} pozitivnih elemenata).

Zadatak

Dokažite da su sljedeće (parcijalno specificirane) funkcije primitivno rekurzivne:

- realizacija „inkrementa" $\mathcal{R}_j$ na kodovima
- realizacija „dekrementa" $\mathcal{R}_j$ (ako je moguć) na kodovima
- $\mathsf{result}(\langle r_0, r_1, r_2, \ldots \rangle) := r_0$ (rezultat izračunavanja)
- $\mathsf{start}(\langle \vec{x} \rangle) := \langle 0, \vec{x}, 0, 0, \ldots \rangle$ (početna konfiguracija)

Praćenje konfiguracija

Neka je I RAM-instrukcija, r i p redom kōd stanja registara i vrijednost programskog brojača prije, a s i q poslije izvršenja I.
Postoje primitivno rekurzivne funkcije $\mathsf{NextReg}^2$ i $\mathsf{NextCount}^3$ takve da za sve takve I, r, p, s i q vrijedi

$$\mathsf{NextReg}(\ulcorner I \urcorner, r) = s \qquad \text{i} \qquad \mathsf{NextCount}(\ulcorner I \urcorner, r, p) = q.$$

Postoje primitivno rekurzivne funkcije Reg^3 i Count^3 takve da, za svaki RAM-program P, za svaki neprazni konačni niz $\vec{x}$,

- $\mathsf{Reg}(\langle \vec{x} \rangle, \ulcorner P \urcorner, n)$ je kod stanja registara, a
- $\mathsf{Count}(\langle \vec{x} \rangle, \ulcorner P \urcorner, n)$ je vrijednost programskog brojača,

nakon n koraka P-izračunavanja s $\vec{x}$.

Relacija $\mathsf{Final}(x, e, n) :\Leftrightarrow$ (x je kod nepraznog konačnog niza $\vec{x}$) $\wedge$ (e je kod RAM-programa P) $\wedge$ (P-izračunavanje s $\vec{x}$ stane nakon *najviše* n koraka), je primitivno rekurzivna.

Brojenje koraka i kodiranje izračunavanja

$\mathsf{Step}(x, e, m) \Longleftrightarrow m = (\mu n \le m)\, \mathsf{Final}(x, e, n)$ znači:
x je kod nepraznog konačnog niza $\vec{x}$, e je kod RAM-programa P, a P-izračunavanje s $\vec{x}$ stane nakon *točno* m koraka.

Čitavo izračunavanje (ako stane) možemo kodirati kao povijest funkcije Reg: $y = \langle \langle r_0^{(s)}, r_1^{(s)}, r_2^{(s)}, \ldots \rangle \rangle_{s \le m}$.

$$\check{T}(x, e, y) :\Longleftrightarrow \mathsf{Step}\big(x, e, \mathsf{pd}(\mathsf{lh}(y))\big) \wedge 1 < y = \overline{\mathsf{Reg}}\big(x, e, \mathsf{lh}(y)\big)$$

Često se relacija $\check{T}$ razdvaja s obzirom na mjesnost ulaza:

$$T_k(\vec{x}^k, e, y) :\Longleftrightarrow \check{T}\big(\mathsf{Code}^k(\vec{x}), e, y\big) \text{ za svaki } k \in \mathbb{N}_+.$$

Zadatak

Dokažite da su relacije Step^3, $\check{T}^3$ i T_k^{k+2} primitivno rekurzivne, imaju funkcijsko svojstvo, te je $\exists_* \mathsf{Step} = \exists_* \check{T}$ [=: HALT].

Funkcija U, univerzalne funkcije comp_k, indeksi

T_k ima *funkcijsko svojstvo*: za sve $\vec{x}$ i e postoji najviše jedan y takav da je $T_k(\vec{x}, e, y)$, i on je parcijalno jednak $\mu z\, T_k(\vec{x}, e, z)$.
Treba nam funkcija koja iz tog y pročita rezultat:

$$U(y) := \mathsf{result}\big(y[\mathsf{pd}(\mathsf{lh}(y))]\big).$$

Tada je $\mathsf{comp}_k := U \circ \mu T_k$ parcijalno rekurzivna funkcija.
Za fiksni $e \in \mathbb{N}$ i $k \in \mathbb{N}_+$ definiramo funkciju *indeksa* e: [ZZC1,2]

$$\{e\}(\vec{x}^k) :\simeq \mathsf{comp}_k(\vec{x}, e).$$

Iz definicije je $\{e\}^k$ parcijalno rekurzivna funkcija, te vrijedi:

Lema (o indeksu)

Neka je $e \in \mathbb{N}$, $k \in \mathbb{N}_+$ te neka je P RAM-program.

- Ako je $e = \ulcorner P \urcorner$, tada P^k računa funkciju $\{e\}^k$.
- Ako $e \notin \mathsf{Prog}$, tada je $\{e\}^k = \otimes^k$.

Kleenejev teorem o normalnoj formi

Teorem (Kleene /kleɪni/)

Postoji primitivno rekurzivna jednomjesna funkcija U takva da za svaki pozitivni prirodni broj (mjesnost) k postoji primitivno rekurzivna $(k+2)$-mjesna relacija T_k tako da za svaku parcijalno rekurzivnu k-mjesnu funkciju F postoji prirodni broj e (*indeks* od F) takav da za svaku k-torku prirodnih brojeva $\vec{x}$ vrijede sljedeće tvrdnje:

- $\vec{x} \in \mathcal{D}_F \Longleftrightarrow \exists y\, T_k(\vec{x}, e, y)$; $\quad [\mathcal{D}_{\{e\}^k} = \$(e, \exists_* T_k)]$
- $F(\vec{x}) \simeq U\big(\mu y\, T_k(\vec{x}, e, y)\big)$.

Korolar

Svaka parcijalno rekurzivna funkcija ima simboličku definiciju s točno jednom pojavom operatora μ.

[Zadatak sa zvjezdicom: Koliko ima pojava operatora $\mathbb{R}$?]

Teorem o grananju, parcijalno rekurzivna verzija

Indeksi nam omogućavaju *metaprogramiranje*: rad s funkcijama pomoću njihovih indeksa. Jedna primjena:

Teorem

Neka su $k, l \in \mathbb{N}_+$, neka su $G_0^k, \ldots, G_l^k$ parcijalno rekurzivne funkcije, te $R_1^k, \ldots, R_l^k$ u parovima disjunktne rekurzivne relacije. Tada su funkcije $F_1^k := \text{if}\{R_1 : G_1, \ldots, R_l : G_l, G_0\}$ i $F_2^k := \text{if}\{R_1 : G_1, \ldots, R_l : G_l\}$ također parcijalno rekurzivne.

Dokaz.

Svaka funkcija G_i ima indeks g_i (kompiliramo je, i kodiramo dobiveni RAM-program). Po rekurzivnoj verziji teorema o grananju $G := \text{if}\{R_1 : C_{g_1}^k, \ldots, R_l : C_{g_l}^k, C_{g_0}^k\}$ je rekurzivna funkcija pa je $F_1 = \text{comp}_k \circ (I_1^k, \ldots, I_k^k, G)$ parcijalno rekurzivna.

Za F_2, samo treba staviti $g_0 := 0$, indeks prazne funkcije $\otimes^k$. □

Turingov stroj

Usprkos tome što je govorio o ljudima kao agentima računanja, Turing je morao uvjeriti svoje čitatelje da su njegove instrukcije sasvim mehaničke i ne zahtijevaju nikakvo eksterno znanje, što je učinio opisavši zamišljeni *stroj* koji ih može provoditi.

Taj stroj je (do na neke tehničke detalje) ono što danas zovemo *Turingovim strojem*. Razlozi zašto ga proučavamo:

- Osmišljen je dovoljno davno da se mnogi drugi sustavi opće izračunljivosti svode na njega (*Turing-potpunost*).
- Motiviran je proučavanjem ljudske kognicije i percepcije.
- Dovoljno je različit od RAM-stroja da će nam dokaz ekvivalencije pokazati veliku robusnost pojma algoritma:
  ono što ljudi rade na papiru nije bitno različito, u principu, od onog što računala rade u siliciju.

RAM-stroj i Turingov stroj su toliko različiti da uopće prirodno ne rade s istim tipom podataka! Ipak, kodiranje spašava stvar.

Turing-izračunavanje

Kažemo da konfiguracija (p, m, s) *prelazi* u konfiguraciju (q, n, t) (po funkciji prijelaza δ) ako, za $(q, \beta, d) := \delta(p, s(m))$, imamo $n = \max\{m + d, 0\}$, $t(m) = \beta$ te $t(j) = s(j)$ za sve $j \neq m$.

Završne konfiguracije (one sa stanjem q_z) prelaze u same sebe.

Definicija

Za riječ $w = \gamma_0\gamma_1\cdots\gamma_{n-1} \in \Sigma^*$, *$w$-trakom* zovemo funkciju $t =: w\textvisiblespace\ldots : \mathbb{N} \to \Sigma_\textvisiblespace := \Sigma \,\dot\cup\, \{\textvisiblespace\} \subseteq \Gamma$ zadanu s $t(j) := \begin{cases} \gamma_j, & j < n \\ \textvisiblespace, & j \geq n \end{cases}$.

Neka je $w \in \Sigma^*$ riječ i $\mathcal{T}$ Turingov stroj s ulaznom abecedom Σ.

$\mathcal{T}$-izračunavanje s w je niz konfiguracijā koji počinje s $(q_0, 0, w\textvisiblespace\ldots)$, te svaka konfiguracija u nizu prelazi u sljedeću.

Kažemo da $\mathcal{T}$-izračunavanje s w *stane* ako u tom nizu postoji (jedinstvena) završna konfiguracija.

Shortlex — kodiranje riječi i jezičnih funkcija

Fiksirajmo neki totalni uređaj $\prec$ na Σ, i proširimo ga na Σ^* tako da riječi poredamo prvo po duljini, a onda leksikografski.

$\varepsilon \prec \boxed{a \prec b \prec c} \prec aa \prec ab \prec ac \prec ba \prec bb \prec bc \prec ca \prec cb \prec cc \prec aaa \prec aab \prec \cdots$

Dokažite: $(\Sigma^*, \prec) \simeq (\mathbb{N}, <)$, sličnost: $\sigma(w) := \text{card}\,\{v \in \Sigma^* \mid v \prec w\}$

Kako brže računati σ i σ^{-1}? Riječ $\leftrightarrow$ zapis u bazi $b' := \text{card}\,\Sigma$.

$$\sigma(\alpha_{n-1}\alpha_{n-2}\cdots\alpha_0) := \sum_{j<n} \sigma(\alpha_j) \cdot (b')^j.$$

Zapis je osebujan jer koristi znamenke iz $[1 \mathinner{..} b']$ umjesto uobičajenih iz $[0 \mathinner{..} b')$, ali sva bitna svojstva i dalje vrijede.

Definicija

Za jezičnu funkciju $\varphi : \Sigma^* \rightharpoonup \Sigma^*$ i fiksirano σ u gornjem smislu, definiramo (jednomjesnu brojevnu) *prateću* funkciju

$$\mathbb{N}\varphi := \sigma \circ \varphi \circ \sigma^{-1}.$$

Povijesni pregled

RAM-stroj je, kao što se pokazalo, dobar za matematički rad s instrukcijama čija minimalnost podsjeća na moderne mikroprocesore arhitekure RISC, no algoritmi su puno stariji od računalā. Prije manje od stoljeća, algoritme su još uvijek izvodili isključivo *ljudi*.

1936. godine Alan Turing je dao jednu od prvih definicija algoritma. Iako je nespretna za strogu matematičku obradu, njena posebnost je u tome što je precizno motivirana detaljnim opisom čovjekova rada i misaonih procesa pri izvođenju algoritma. Turingov članak time predstavlja najbolju moguću formalizaciju pojma algoritma kako su ga ljudi shvaćali oduvijek, sve do druge polovice prošlog stoljeća.

Jezični model izračunavanja

Definicija

Abeceda Σ je (fiksiran) konačni neprazni skup.
Znak α je proizvoljni element abecede.
Riječ w je konačan niz znakova; može biti i prazna (oznaka ε).
Skup svih riječi označavamo sa Σ^*. *Jezik* znači podskup od Σ^*.
Jezična funkcija φ je bilo koja parcijalna funkcija iz Σ^* u Σ^*.
Turingov stroj $\mathcal{T} = (Q, \Sigma, \Gamma, \textvisiblespace, \delta, q_0, q_z)$ zadan je pomoću:

- konačnog skupa stanja Q, s istaknutim elementima q_0 (*početno stanje*) i q_z (*završno stanje*);
- konačne radne abecede Γ koja je nadskup *ulazne abecede* Σ s istaknutim elementom $\textvisiblespace \in \Gamma \setminus \Sigma$ (*praznina*); $\Gamma_+ := \Gamma \setminus \{\textvisiblespace\}$;
- funkcije prijelaza $\delta : (Q \setminus \{q_z\}) \times \Gamma \to Q \times \Gamma \times \{-1, 1\}$.

Konfiguracija Turingova stroja zadana je *stanjem* $q \in Q$, *pozicijom* $n \in \mathbb{N}$ i *trakom* $t : \mathbb{N} \to \Gamma$ s konačnim nosačem $t^{-1}[\Gamma_+]$.

Turing-izračunljivost jezičnih funkcija

Definicija

Neka je $\varphi : \Sigma^* \rightharpoonup \Sigma^*$ jezična funkcija (dakle $\mathcal{D}_\varphi \subseteq \Sigma^*$), te $\mathcal{T}$ Turingov stroj čija ulazna abeceda je Σ.

Kažemo da $\mathcal{T}$ *računa* φ ako za svaku riječ $w \in \Sigma^*$ vrijedi:

- $w \in \mathcal{D}_\varphi$ ako i samo ako $\mathcal{T}$-izračunavanje s w stane, te
- tada, traka završne konfiguracije je upravo $\varphi(w)$-traka.

Za jezičnu funkciju φ kažemo da je *Turing-izračunljiva* ako postoji Turingov stroj koji je računa.

Za razliku od RAM-strojeva, ne računa svaki Turingov stroj neku jezičnu funkciju nad svojom abecedom: ako za bilo koju $w \in \Sigma^*$, u završnoj konfiguraciji traka nije v-traka ni za koju $v \in \Sigma^*$ (jer su na njoj ostali neki znakovi iz $\Gamma_+ \setminus \Sigma$, ili pak postoji znak iz Σ nakon prve praznine), taj Turingov stroj ne računa nijednu jezičnu funkciju.

Vježbanje rada s pomaknutom bazom

Zadatak

Neka je $\Sigma = \{a, b, c\}$ uređena abecedno.

- Napišite prvih 15 elemenata od Σ^* redom po kodovima.
- Odredite kōd riječi baca.
- Odredite riječ koja ima kōd 253.

Promotrimo jezičnu funkciju $\xi : \Sigma^* \rightharpoonup \Sigma^*$ koja zamjenjuje posljednju pojavu znaka c u riječi znakom b.

- Odredite prirodnu domenu od ξ.
- Odredite (ako postoje) $\mathbb{N}\xi(253)$ i $\mathbb{N}\xi(70)$.
- Nacrtajte dijagram Turingova stroja koji računa ξ.

Za funkciju $\alpha(w) := wa$, napišite RAM-program koji računa $\mathbb{N}\alpha$.

Sistematičniji rad s pomaknutom bazom

Zadatak

Nađite primitivno rekurzivne funkcije koje zadovoljavaju ove parcijalne specifikacije (zapisi su u pomaknutoj bazi $b \in \mathbb{N}_+$):

- $\mathsf{nw}(t, b) :=$ ukupan broj zapisa riječi duljine $\leq t$ $[:= \sum_{i \leq t} b^i]$
- $\mathsf{slh}(n, b) :=$ duljina zapisa od n $[:= (\mu t \leq n)(\mathsf{nw}(t, b) > n)]$
- $m \mathbin{\wr} n :=$ (broj čiji je zapis) konkatenacija (zapisa) od m i n
- $\mathsf{sprefix}(n, i, b) :=$ početak zapisa od n duljine i $(\leq \mathsf{slh}(n, b))$
- $\mathsf{mod}'(n, b) :=$ pomaknuti ostatak n pri dijeljenju s b
- $\mathsf{sdigit}(n, i, b) :=$ i-ta znamenka zapisa n slijeva, $i < \mathsf{slh}(n, b)$

Dakle, σ je kodiranje skupa Σ^*. Zato pišemo $\sigma =: \mathbb{N}\Sigma^*$, $\sigma|_\Sigma =: \mathbb{N}\Sigma$.
[Zadani $\mathbb{N}\Sigma$ proširujemo do $\mathbb{N}\Sigma^*$ zapisom u pomaknutoj bazi b'.]
Ako se $\mathbb{N}\Sigma$ podrazumijeva, $\sigma(w)$ još pišemo $\langle w \rangle$.

Kodiranje radne abecede i trake

Do daljnjeg, neka je $\mathcal{T} = (Q, \Sigma, \Gamma, \sqcup, \delta, q_0, q_z)$ fiksni Turingov stroj, te $\mathbb{N}\Sigma$ fiksno kodiranje njegove ulazne abecede.

Označimo $b := \operatorname{card} \Gamma > b'$. Proširimo $\mathbb{N}\Sigma$ do $\mathbb{N}\Gamma$ tako da preslikamo $\sqcup$ u 0, a znakove iz $\Gamma_+ \setminus \Sigma$ bijektivno u $\langle b' \cdots b \rangle$.

Sada proširimo $\mathbb{N}\Gamma$ do kodiranja proizvoljne trake:

$$\langle t \rangle := \langle t_0 t_1 \ldots \sqcup\sqcup \ldots \rangle := \sum_{j \in \mathbb{N}} \mathbb{N}\Gamma(t_j) \cdot b^j.$$

Lema

Postoji primitivno rekurzivna funkcija Recode^3 takva da za svaku riječ $w \in \Sigma^*$ vrijede jednakosti

$$\mathsf{Recode}(\langle w \rangle, b', b) = \langle w \sqcup \ldots \rangle,$$
$$\mathsf{Recode}(\langle w \sqcup \ldots \rangle, b, b') = \langle w \rangle.$$

Kodiranje stanjā i funkcije prijelaza

Označimo $a := \operatorname{card} Q$ i fiksirajmo bijekciju $\mathbb{N}Q : Q \leftrightarrow [0 \cdots a\rangle$ takvu da je $\mathbb{N}Q(q_0) = 0$ i $\mathbb{N}Q(q_z) = 1$ (BSOMP $q_0 \neq q_z$).

Definiramo tri konačne funkcije s domenom $D := [0 \cdots a\rangle \times [0 \cdots b\rangle$ i kodomenama redom $[0 \cdots a\rangle$, $[0 \cdots b\rangle$ i $\{0, 1, 2\}$, tako da za sve $(s, g) \in D$ označimo $(q, \gamma, d) := \delta(\mathbb{N}Q^{-1}(s), \mathbb{N}\Gamma^{-1}(g))$, i stavimo

$$\mathsf{newstate}(s, g) := \mathbb{N}Q(q) \qquad \mathsf{newstate}(e, s, g) := e[s][g][0]$$
$$\mathsf{newsymbol}(s, g) := \mathbb{N}\Gamma(\gamma) \qquad \mathsf{newsymbol}(e, s, g) := e[s][g][1]$$
$$\mathsf{direction}(s, g) := 1 + d \qquad \mathsf{direction}(e, s, g) := e[s][g][2]$$

To ne radi za $s = 1$ (jer δ nije definirana za stanje q_z); tada je

$$\mathsf{newstate}(1, g) := 1, \quad \mathsf{newsymbol}(1, g) := g, \quad \mathsf{direction}(1, g) := 1.$$

Kako su to konačne funkcije, njihova proširenja nulom (koja zovemo istim imenima) su funkcije s konačnim nosačem, pa su primitivno rekurzivne (dobivene editiranjem nulfunkcije C_0^2).

Primjer kodiranja funkcije prijelaza

Napisali ste funkciju prijelaza za Turingov stroj koji računa jezičnu funkciju ξ, koja posljednje c u riječi zamjenjuje s b.

Odgovarajuće brojevne tablice su

	newstate					newsymbol					direction			
	Γ	$\sqcup$	a	b	c		$\sqcup$	a	b	c		$\sqcup$	a	b	c
Q	$\mathbb{N}\Gamma$	0	1	2	3		0	1	2	3		0	1	2	3
q_0	0	3	0	0	0	0	0	1	2	3	0	0	2	2	2
q_z	1	1	1	1	1	1	0	1	2	3	1	1	1	1	1
q_x	2	2	2	2	2	2	0	1	2	3	2	0	0	0	0
q_L	3	2	3	3	1	3	0	1	2	2	3	0	0	0	2

i tako možemo kodirati Turingov stroj kao 3D *array* $a \times b \times 3$:

$$\ulcorner \mathcal{T}_\xi \urcorner := \langle \langle \langle 3, 0, 0 \rangle, \langle 0, 1, 2 \rangle, \langle 0, 2, 2 \rangle, \langle 0, 3, 2 \rangle \rangle, \langle \langle 1, 0, 1 \rangle, \ldots \rangle, \ldots \rangle.$$

Funkcije s prethodnog slajda: dvomjesne za fiksni Turingov stroj, a tromjesne za bilo koji Turingov stroj zadan kodom.

Kodiranje $\mathcal{T}$-izračunavanja

Lema (2 komada)

$$\mathsf{State}(\ulcorner \mathcal{T} \urcorner, \langle w \rangle, n) := \mathbb{N}Q(q_n)$$
$$\mathsf{Position}(\ulcorner \mathcal{T} \urcorner, \langle w \rangle, n) := p_n$$
$$\mathsf{Tape}(\ulcorner \mathcal{T} \urcorner, \langle w \rangle, n) := \langle t_n \rangle$$

gdje je $((q_n, p_n, t_n))_{n \in \mathbb{N}}$ $\mathcal{T}$-izračunavanje s $w \in \Sigma^*$ (a $\mathcal{T}$ je Turingov stroj nad Σ) su primitivno rekurzivne funkcije.

Dokaz: primjena simultane primitivne rekurzije.

$$G_1(e, x) := G_2(e, x) := 0, \quad G_3(e, x) := \mathsf{Recode}(x, b', b).$$
$$H_1(e, x, n, q, p, t) := \mathsf{newstate}(e, q, g),$$
$$H_2(e, x, n, q, p, t) := \mathsf{pd}(p + \mathsf{direction}(e, q, g)),$$
$$H_3(e, x, n, q, p, t) := t \dot{-} g \cdot b^p + \mathsf{newsymbol}(e, q, g) \cdot b^p,$$

uz pokrate $g := t \mathbin{/\!/} b^p \bmod b$, $b' := \operatorname{card} \Sigma$, $b := \mathsf{lh}(e[0])$. □

Zaustavljanje i čitanje rezultata

Parcijalno rekurzivna funkcija $\mathsf{stop}(e, x) :\simeq \mu n(\mathsf{State}(e, x, n) = 1)$: nakon koliko koraka $\mathcal{T}$-izračunavanje s w stane; $e = \ulcorner \mathcal{T} \urcorner$, $x = \langle w \rangle$.

Teorem ($t \Rightarrow p$ za jezične funkcije)

Neka je Σ abeceda, $\mathbb{N}\Sigma$ njeno kodiranje, i $\varphi : \Sigma^* \rightharpoonup \Sigma^*$ jezična funkcija nad tom abecedom. Ako je φ Turing-izračunljiva, tada je $\mathbb{N}\varphi$ parcijalno rekurzivna.

Dokaz.

Po pretpostavci postoji Turingov stroj $\mathcal{T}$ koji računa φ.
[Označimo $e := \ulcorner \mathcal{T} \urcorner$, $b := \mathsf{lh}(e[0])$, $b' := \operatorname{card} \Sigma$.]
Primjenom postupka s prethodnih slajdova na $\mathcal{T}$, dobivamo

$$\mathbb{N}\varphi(x) \simeq \mathsf{Recode}(\mathsf{Tape}(e, x, \mathsf{stop}(e, x)), b, b')$$

pa je $\mathbb{N}\varphi$ parcijalno rekurzivna kao kompozicija takvih. □

Transpiler RAM $\rightsquigarrow$ Turing: prvi i posljednji fragment

Neka RAM-algoritam P^1 širine m računa $\mathbb{N}\varphi$.
Želimo izgraditi Turingov stroj za φ.

U Γ (uz Σ) dodajmo elemente # (separator), \$ (graničnik), i još sve elemente skupa $B^m := \{\circ, \bullet\}^m$, koje promatramo kao stupce: dva specijalna su $\sqcup := (\circ, \circ, \circ, \circ, \ldots, \circ)^\top$ i $r_1 := (\circ, \bullet, \circ, \circ, \ldots, \circ)^\top$.

Graničnik pokazuje početak i kraj (korištenog dijela) trake, a separator odvaja ulaz (Σ-dio) od simulacije (B^m-dio).

Označimo $b := \operatorname{card} \Sigma$, i za $t \in [1 \cdots b]$ označimo $\alpha_t := \mathbb{N}\Sigma^{-1}(t)$.

Zadatak

Konstruirajte fragmente koji prevode konfiguracije:

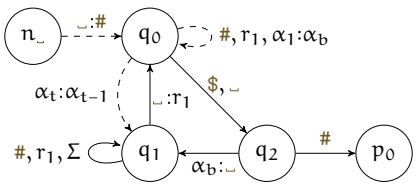
- (F1): $(n_\$, 0, w\sqcup \ldots)$ (početna s ulazom w)
  u $(n_\sqcup, |w| + 1, \$w\sqcup \ldots)$
- (F5): $(l_\sqcup, |v|, \$v\sqcup \ldots)$ u $(l_\$, 0, v\sqcup \ldots)$ (završna s izlazom v)

Drugi fragment (F2) — inicijalizacija RAM-stroja

Cilj: prevesti konfiguraciju $\underbrace{\$ \; w \; \sqcup \ldots}_{n_\sqcup}$ (izlaz prvog fragmenta) u konfiguraciju $\underbrace{\$ \; \sqcup|w| \; \# \; r_1{}^{\langle w \rangle} \; \sqcup \ldots}_{p_0}$.

Metoda: dopišemo #, dekrementiramo broj u pomaknutoj bazi b te za svaki uspješni dekrement dodamo po jedan r_1 na kraj.

Implementacija:



Registri kao tragovi trake

F1 i F2 uopće ne ovise o RAM-programu P (osim kroz broj m), već samo o ulazu w i kodiranju $\mathbb{N}\Sigma$. Taj dio sigurno stane.

F3 simulira P-izračunavanje s $\langle w\rangle$ (na dijelu trake od pozicije znaka #; označimo je s k).

Sadržaj registara zapisujemo u „retke" (*tragove*) trake: gledano od # prema desno, j-ti redak brojeći od 0 odozgo uvijek će biti oblika $\bullet\bullet\cdots\bullet\circ\circ\cdots$, gdje je broj kružića $\bullet$ sadržaj registra $\mathcal{R}_j$.

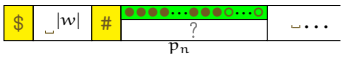
Za $\beta, \beta' \in B := \{\circ, \bullet\}$, za $j \in [0 \cdot\cdot\, m\rangle$, za $q, q' \in Q$ te za $d \in \{-1, 1\}$,

$$\delta_{(j)}(q,\beta) := (q', \beta', d) \quad \text{(grafički prikaz: } q \xrightarrow{\beta:\beta'@j} q' \text{ za } d = 1\text{)}$$

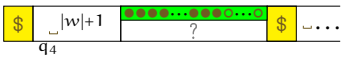
nam je pokrata za 2^{m-1} prijelaza oblika $\delta(q,c) := (q', c', d)$, gdje se $c, c' \in B^m$ podudaraju na svim koordinatama osim eventualno j-te, gdje u c piše β a u c' piše β'.

Četvrti fragment (F4) — dekodiranje izlaza

Cilj: prevesti konfiguraciju

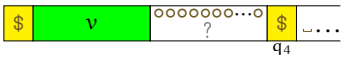


(u gornjem redu je $\bullet^{(v)}$) u



,

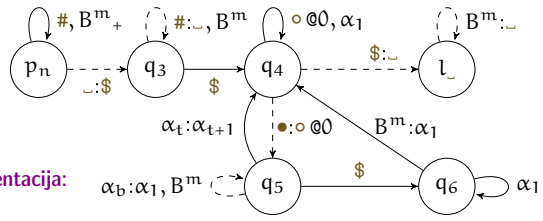
a zatim (metoda dolje↓) u



te obrisati ove ? i zadnji $ (namjestiti konfiguraciju za F5).

Metoda: obrnuto od F2; brišemo znakove $\bullet$ iz traga 0 slijeva te za svaki (od y njih) maknuti $\bullet$ inkrementiramo riječ prije $.

Implementacija:



Gadgets za pojedine instrukcije; treći fragment (F3)

Neka su $I_0, \ldots, I_{n-1}$ sve instrukcije od P redom. Za svaku ćemo imati stanja p_i i s_i, a za instrukcije tipa DEC još i stanje t_i.

Stanja $p_i, i < n$ pomiču glavu s bilo koje pozicije $\geq k$ na $k+1$.

Zadatak (4 komada)

Implementirajte funkcionalnost stanja p_i (za sve $i < n$) te stanja s_i (i eventualno t_i) ovisno o tipu instrukcije I_i.

Ako/kad P-izračunavanje s $\langle w\rangle$ stane, Turingov stroj koji upravo gradimo doći će u stanje p_n, i u najvišem tragu nakon pozicije k pisat će $\bullet^y\circ\cdots$, za $y := \mathbb{N}\varphi(\langle w\rangle) = \langle\varphi(w)\rangle$.

Ako to izračunavanje ne stane, stroj neće nikad doći u p_n — a svaki put od početnog do završnog stanja vodi preko p_n.

RAM-izračunljivost povlači Turing-izračunljivost

Teorem ($p \Rightarrow t$ za jezične funkcije)

Neka je Σ abeceda, $\mathbb{N}\Sigma$ njeno kodiranje, te $\varphi : \Sigma^* \rightharpoonup \Sigma^*$ jezična funkcija nad tom abecedom. Ako je $\mathbb{N}\varphi$ parcijalno rekurzivna, tada je φ Turing-izračunljiva.

Dokaz.

Upravo smo, koristeći RAM-program P koji računa $\mathbb{N}\varphi$ (koji dobijemo kompiliranjem simboličke definicije od $\mathbb{N}\varphi$), konstruirali Turingov stroj koji računa φ, tako da za ulaz w:

- nakon $ i w zapiše # i $x := \langle w\rangle$ u unarnom zapisu $(r_1{}^x)$;
- na dijelu trake nakon # simulira P-izračunavanje s $x = \langle w\rangle$;
- dekodira rezultat $y := \mathbb{N}\varphi(\langle w\rangle) = \langle\varphi(w)\rangle$ u riječ $v := \varphi(w)$;
- čisti ostatak trake i premješta $v = \varphi(w)$ na početak. □

Izračunljivost jezikā

Kao što smo uz brojevne funkcije gledali i brojevne *relacije* (podskupove od $\mathbb{N}^k$), možemo uz jezične funkcije gledati *jezike* (podskupove od Σ^*).

Definicija

Neka je Σ abeceda i $\mathbb{N}\Sigma$ neko njeno kodiranje. Za jezik $L \subseteq \Sigma^*$ kažemo da ima neko svojstvo izračunljivosti ako relacija $\langle L\rangle := \sigma[L] \subseteq \mathbb{N}$ (odnosno funkcija $\chi_{\langle L\rangle}$) ima to svojstvo.

Zadatak (veza s kolegijem INTERPRETACIJA PROGRAMA)

Dokažite da je definicija rekurzivnog jezika u gornjem smislu ($\langle L\rangle$ je rekurzivna) ekvivalentna definiciji iz INTERPRETACIJE PROGRAMA (postoji Turingov odlučitelj koji prepoznaje L).

Uputa: ($\Leftarrow$) malo doradite funkcije stop i „$\mathbb{N}\varphi$" za odlučitelje; ($\Rightarrow$) dodajte 2 prijelaza iz $l_\$$ transpiliranog stroja za $\chi_{\langle L\rangle}$. ZZD2

Binarni zapis više argumenata brojevne funkcije

Definicija

Binarna abeceda je $\Sigma_\beta := \{\bullet, /\}$: sastoji se od *kružića* $\bullet$ i *separatora* /. Fiksirajmo poredak $\bullet \prec /$, odnosno kodove

$$\langle\bullet\rangle := 1, \qquad \langle/\rangle := 2 \qquad \text{(pomaknuta baza 2).}$$

Definiramo primitivno rekurzivnu funkciju $\mathsf{blh}(n) := \mathsf{slh}(n, 2)$.

Za $k \in \mathbb{N}_+$, *binarni zapis* k-torke $\vec{x} = (x_1, \ldots, x_k) \in \mathbb{N}^k$ je riječ

$$\beta(\vec{x}) := \bullet^{x_1}/\bullet^{x_2}/\cdots/\bullet^{x_k} \in \Sigma_\beta^*.$$

Zadatak

Dokažite da je β bijekcija između $\mathbb{N}^+$ i Σ_β^*. Opišite inverz β^{-1}.

Binarna reprezentacija brojevne funkcije

Definicija

Za brojevnu funkciju f^k definiramo *binarnu reprezentaciju* βf:

$$\beta f : \Sigma_\beta^* \rightharpoonup \Sigma_\beta^*, \quad \beta f(w) :\simeq \beta\big(f(\vec{x})\big), \text{ ako je } w = \beta(\vec{x}^k).$$

Jezična funkcija βf dobro modelira sve aspekte izračunljivosti brojevne funkcije f. Zato skraćeno kažemo da je f *Turing-izračunljiva* ako je βf Turing-izračunljiva.

Primjenom dokazanih rezultata za jezične funkcije na βf možemo dobiti analogne rezultate za brojevne funkcije — no prvo trebamo nekoliko tehničkih lema.

Primitivno rekurzivne funkcije bin^k

Lema

Za svaki $k \in \mathbb{N}_+$, $\mathsf{bin}^k := \mathbb{N}\Sigma_\beta^* \circ \beta^k$ je primitivno rekurzivna.

Dokaz.

Indukcijom po k. Baza je formula za sumu geometrijskog niza:

$$\mathsf{bin}^1(x) = \langle\bullet^x\rangle = (11\cdots1)_2 = 2^x \dot{-} 1 = \mathsf{pd}\big(\mathsf{pow}(2, x)\big).$$

Pretpostavimo da je bin^l primitivno rekurzivna za neki $l \in \mathbb{N}_+$.

Korak: Tada je $\mathsf{bin}^{l+1}(\vec{x}, y) = \langle\bullet^{x_1}/\cdots/\bullet^{x_l}/\bullet^y\rangle =$
$= \langle\bullet^{x_1}/\cdots/\bullet^{x_l}\rangle \mathbin{\hat{2}} \langle/\rangle \mathbin{\hat{2}} \langle\bullet^y\rangle = \mathsf{bin}^l(\vec{x}) \mathbin{\hat{2}} 2 \mathbin{\hat{2}} \mathsf{bin}^1(y)$,

što je primitivno rekurzivno kao kompozicija od bin^l, bin^1, konstante 2, operacije $\hat{2}$ i koordinatnih projekcijā. □

Lovljenje strelicā po komutativnim dijagramima

Zadatak

Dokažite $\mathsf{blh} \circ \mathsf{bin}^1 = \mathsf{I}^1_1$. [Podsjetnik: $\mathsf{blh}(n) := \mathsf{slh}(n, 2)$.]

Propozicija (Dijagram)

Za svaki $k \in \mathbb{N}_+$, za svaku brojevnu funkciju f^k vrijedi

$$\mathsf{bin}^1 \circ f = \mathbb{N}\beta f \circ \mathsf{bin}^k.$$

$$\begin{tikzcd}
\mathbb{N}^k \arrow[r, "f"] \arrow[d, "\beta^k"] \arrow[dd, bend right=60, "\mathsf{bin}^k"'] & \mathbb{N} \arrow[d, "\beta^1"] \arrow[dd, bend left=60, "\mathsf{bin}^1"] \\
\Sigma^*_\beta \arrow[r, "\beta f"] \arrow[d, "\mathbb{N}\Sigma^*_\beta"] & \bullet^* \arrow[d, "\mathbb{N}\Sigma^*_\beta"] \\
\mathcal{J}^1_{\mathsf{bin}^k} \arrow[r, "\mathbb{N}\beta f"] \arrow[uu, bend left=60, "\mathsf{arg}_1, \ldots, \mathsf{arg}_k"] & \mathcal{J}^1_{\mathsf{bin}^1} \arrow[uu, bend right=60, "\mathsf{blh}^1"']
\end{tikzcd}$$

Funkcije $\mathsf{arg}_1, \ldots, \mathsf{arg}_k$

Zadatak

Dokažite primitivnu rekurzivnost sljedećih relacija i funkcija:

- $\langle v \rangle \sqsubseteq \langle w \rangle$: v je prefiks od w
- $\mathsf{count2}(x)$: broj dvojki u (zapisu od) x
- $\mathcal{J}_{\mathsf{bin}^k} = \langle \beta[\mathbb{N}^k] \rangle$: slika k-mjesnog binarnog kodiranja
- $z := \hat{x}$: uokvirivanje (zapisa od) x dvojkama: $\hat{x} = (2x2)_2$
- $\mathsf{pos2}(z, i) :=$ pozicija i-te dvojke u (zapisu uokvirenog) z
- $\mathsf{streak1}(z, i) :=$ duljina i-tog niza uzastopnih jedinica u z
- $\mathsf{arg}_n(\mathsf{bin}^k(\vec{x})) = \langle \vec{x}^k \rangle[n-1]$ $\big(= \mathsf{I}^k_n(\vec{x})$ za $1 \le n \le k\big)$

Pod zapisom podrazumijevamo zapis u pomaknutoj bazi 2.
i broji od 0 slijeva; za prevelik i $\mathsf{pos2}$ daje blh, a $\mathsf{streak1}$ daje 0.

Jedan odgovor: $\mathsf{pos2}(z, i) := \mathsf{blh}\big((\mu y < z)(\hat{y} \sqsubseteq \hat{z} \wedge \mathsf{count2}(y) = i)\big)$

Teorem ekvivalencije

Teorem

Neka je $k \in \mathbb{N}_+$, te f^k funkcija. Tada je ekvivalentno:

(p) f je parcijalno rekurzivna (dobivena s $\circ$, $\mathbb{R}$, μ iz Z, Sc, I^k_n)
(s) f ima simboličku definiciju (f je u korijenu stabla ...)
(r) f je RAM-izračunljiva (neki RAM-algoritam računa f)
(m) f je makro-izračunljiva (neki makro-algoritam računa f)
(i) f ima indeks (postoji $e \in \mathbb{N}$ takav da je $f = \{e\}^k$)
(t) f je Turing-izračunljiva (neki Turingov stroj računa βf)

algoritam / model	konkretni (imperativni)	apstraktni (funkcijski)
jezični (praktični)	Turingov stroj	λ-račun
brojevni (teorijski)	RAM-stroj	rekurzivnost

Russellova funkcija i Kleenejev skup

Definicija

Definiramo $\mathsf{Russell}(x) :\simeq 1 + \mathsf{comp}(x, x)$. Označimo $K^1 := \mathcal{D}_{\mathsf{Russell}}$.

$\mathsf{Russell} = \mathsf{Sc} \circ \mathsf{U} \circ \mu\mathsf{T}_1 \circ (\mathsf{I}^1_1, \mathsf{I}^1_1)$ je parcijalno rekurzivna. Ali ...

Lema

Ne postoji rekurzivna funkcija r takva da je $r|_K = \mathsf{Russell}$.

Dokaz.

Pretpostavimo da postoji. Tada r ima indeks, označimo jedan takav s e. Kako je r totalna, postoji $q := r(e) \in \mathbb{N}$. No tada

$$\mathbb{N} \ni 1 + q = 1 + r(e) = 1 + \{e\}(e) = 1 + \mathsf{comp}(e, e)$$

pokazuje da je $e \in K$, pa je $1 + q = \mathsf{Russell}(e) \neq r(e) = q$.
Dakle r ne može biti proširenje od $\mathsf{Russell}$. □

Turing-izračunljivost povlači parcijalnu rekurzivnost

Teorem ($t \Rightarrow p$ za brojevne funkcije)

Neka je $k \in \mathbb{N}_+$ i f^k (brojevna) funkcija.
Ako je βf Turing-izračunljiva, tada je f parcijalno rekurzivna.

Dokaz.

Po teoremu [$t \Rightarrow p$ za jezične funkcije],
$\mathbb{N}\beta f$ je parcijalno rekurzivna. Iz dijagrama slijedi da je

$$f = \mathsf{blh} \circ \mathbb{N}\beta f \circ \mathsf{bin}^k,$$

dakle parcijalno rekurzivna kao kompozicija triju takvih. □

Obrat je kompliciraniji: umjesto lijevog inverza blh za bin^1, sada trebamo k funkcijā $(\mathsf{arg}_i)_{i=1}^k$ koje čine desni inverz za bin^k.

Turing-izračunljivost parcijalno rekurzivnih funkcija

Teorem ($p \Rightarrow t$ za brojevne funkcije)

Binarna reprezentacija svake parcijalno rekurzivne brojevne funkcije je Turing-izračunljiva.

Dokaz.

Neka je $k \in \mathbb{N}_+$ i f^k parcijalno rekurzivna. Iz dijagrama slijedi

$$\mathbb{N}\beta f = \big(\mathsf{bin}^1 \circ f \circ (\mathsf{arg}_1, \ldots, \mathsf{arg}_k)\big)\big|_{\mathcal{J}_{\mathsf{bin}^k}}$$

što je parcijalno rekurzivna funkcija kao kompozicija takvih, restringirana na rekurzivan skup.
Iz [$p \Rightarrow t$ za jezične funkcije], βf je Turing-izračunljiva. □

Sustavi opće izračunljivosti i univerzalnost

Definirali smo četiri formalizma izračunljivosti i pokazali da su ekvivalentni: računaju iste funkcije ako rade s istim tipom podataka, inače „iste" do na neko prirodno kodiranje.

Štoviše, jedan od njih (pa onda i svi ostali) je univerzalan: familija parcijalno rekurzivnih funkcija comp_k može računati *bilo koju* parcijalno rekurzivnu funkciju zadanu indeksom.

U zadnjih stotinjak godina predloženi su brojni formalizmi (opće gramatike, λ-izrazi, ...) i **svi su se pokazali ekvivalentnima**.
Church–Turingova teza tvrdi da postoji intuitivna ideja izračunljivosti koju svi ti formalizmi uspješno modeliraju.

Ipak, postoje i *neizračunljive* funkcije, koje čak ni univerzalni sustavi ne mogu računati. To ćemo dokazati dijagonalizacijom.

Nerekurzivna (totalna) funkcija i relacija

Korolar

Funkcija zadana s $\widehat{\mathsf{Russell}}(x) := \begin{cases} \mathsf{Russell}(x), & K(x) \\ 0, & \text{inače} \end{cases}$
je totalna ali nije rekurzivna. Kleenejev skup K nije rekurzivan.

Dokaz.

Kad bi $\widehat{\mathsf{Russell}}$ bila rekurzivna, tada bi ona bila jedna od funkcijā r za koje prethodna lema tvrdi da ne mogu postojati.
A kad bi K bio rekurzivan, tada bi i $\widehat{\mathsf{Russell}} = \mathsf{if}\{K : \mathsf{Russell}, Z\}$ bila parcijalno rekurzivna po teoremu o grananju (parcijalno rekurzivna verzija). □

[Primijetimo da smo time dobili još jedan primjer totalne nerekurzivne funkcije: konkretno, χ_K.]

Problem zaustavljanja

Po teoremu ekvivalencije,
postoje RAM-stroj i Turingov stroj koji računaju Russell.

Kad bismo za bilo koji od ta dva stroja imali efektivni postupak za određivanje stane li s proizvoljnim ulazom, po Church–Turingovoj tezi taj efektivni postupak bi računao rekurzivnu funkciju. No kako χ_K nije rekurzivna, proizlazi da takav efektivni postupak ne može postojati.

[Formalno, Turingov stroj računa βRussell $=: \rho$, no iz definicije β slijedi da za svaki $t \in \mathbb{N}$ vrijedi: $\bullet^t \in \mathcal{D}_\rho$ ako i samo ako je $t \in K$.]

Korolar

Problem zaustavljanja (*halting problem*)
za Turingove strojeve i za RAM-strojeve je neodlučiv.

Churchov teorem — uvođenje oznakā

Kompilirajmo funkciju Russell u RAM-program P.
Označimo s m i n redom širinu i duljinu algoritma P^1.

Signatura σ nad kojom ćemo konstruirati formule ima:

- jedan konstantski simbol o
- jedan jednomjesni funkcijski simbol s
- $n+1$ m-mjesnih relacijskih simbola $R_0, \ldots, R_n$.

Od individualnih varijabli koristit ćemo samo $x_j, j < m$.

[Intendirana semantika: zatvoreni termi očito predstavljaju prirodne brojeve. Atomarne formule $R_i(x_0, \ldots, x_{m-1})$ predstavljaju dostižne konfiguracije RAM-stroja: i je vrijednost od PC, a valuacija od x_j je sadržaj registra $\mathcal{R}_j$.]

Početna i završna formula

Za svaki $t \in \mathbb{N}$, definiramo

$$\pi_t := R_0\big(o, \overbrace{s(s(\cdots s(o)\cdots))}^{t\text{ s-ova}}, \overbrace{o, o, \ldots, o}^{m-2\text{ o-ova}}\big),$$
$$\Gamma_t := \{\pi_t, \iota_0, \iota_1, \iota_2, \ldots, \iota_{n-1}\},$$
$$\zeta := \exists x_0 \exists x_1 \cdots \exists x_{m-1} R_n(x_0, x_1, \ldots, x_{m-1}).$$

Zadatak

Generalizirajte sve na ovom slajdu na proizvoljni RAM-algoritam P^k i ulazne podatke za njega $\vec{t}^k$.

Lema

Za svaki $t \in \mathbb{N}$ vrijedi: (P-izračunavanje s t stane) $\iff (\Gamma_t \vDash \zeta)$.

Dokaz leme, smjer ($\Rightarrow$)

Pretpostavimo sada da P-izračunavanje s t stane, i označimo broj koraka u tom izračunavanju s b. Kao i u dokazu obrnutog smjera, uvedimo oznake $r_j^{(k)}$ i $\text{PC}^{(k)}$ (samo sada za $k \le b$).

Neka je $\mathfrak{M} = \big(M, \Theta, f, (S_i)_{i \le n}\big)$ proizvoljna σ-struktura u kojoj vrijedi Γ_t. Označimo s $a_j^{(k)} := \overline{r_j^{(k)}}^{\mathfrak{M}} = f(f(\cdots f(\Theta)\cdots)) \in M$ element nosača dobiven $r_j^{(k)}$-strukom primjenom f na Θ.

Indukcijom po k dokažemo da za svaki $k \le b$ vrijedi

$$\vec{a}_k := \big(a_0^{(k)}, a_1^{(k)}, \ldots, a_{m-1}^{(k)}\big) \in S_{\text{PC}^{(k)}}.$$

Baza slijedi iz $\mathfrak{M} \vDash \pi_t$, a korak iz $\mathfrak{M} \vDash \iota_{\text{PC}^{(k)}}$.

Za $k := b$ dobijemo $\vec{a}_b \in S_{\text{PC}^{(b)}} = S_n \neq \varnothing$, dakle $\mathfrak{M} \vDash \zeta$.

Prema Churchovu teoremu

Prethodni slajd pokazuje put k dokazu Churchova teorema.

Ideja je za svaki prirodni broj t konstruirati formulu prvog reda, tako da t bude u K ako i samo ako je ta formula valjana.

Formula će modelirati problem zaključivanja (logičke posljedice)

$$\pi_t, \iota_0, \iota_1, \ldots, \iota_{n-1} \vDash^{?} \zeta$$

gdje jedino π_t ovisi o t, dok su sve ostale formule, kao i broj n, fiksne. Taj problem možemo prikazati kao problem utvrđivanja valjanosti formule φ_t oblika

$$\pi_t \to \big(\overline{\iota_0 \wedge \iota_1 \wedge \cdots \wedge \iota_{n-1}} \to \zeta\big)$$

(gdje $\overline{\psi}$ označava univerzalno zatvorenje formule ψ, dok su π_t i ζ rečenice), iz čega će slijediti tvrdnja Churchova teorema.

Formule ι_i

Za $i < n$, definiramo formulu ι_i ovisno o i-toj instrukciji od P:

- Ako je to (i. INC $\mathcal{R}_j$), definiramo ι_i kao formulu
$$R_i\big(x_{[0..m)}\big) \to R_{i+1}\big(x_{[0..j)}, s(x_j), x_{(j..m)}\big).$$
- Ako je to (i. DEC $\mathcal{R}_j, l$), definiramo ι_i kao formulu
$$\big(R_i\big(x_{[0..j)}, o, x_{(j..m)}\big) \to R_l\big(x_{[0..j)}, o, x_{(j..m)}\big)\big) \wedge$$
$$\wedge \big(R_i\big(x_{[0..j)}, s(x_j), x_{(j..m)}\big) \to R_{i+1}\big(x_{[0..m)}\big)\big).$$
- Ako je to (i. GO TO l), definiramo ι_i kao formulu
$$R_i\big(x_{[0..m)}\big) \to R_l\big(x_{[0..m)}\big).$$

Dokaz leme, smjer ($\Leftarrow$)

Neka je $t \in \mathbb{N}$ proizvoljan, i ζ logička posljedica od Γ_t.

Pretpostavimo suprotno da P-izračunavanje s t ne stane, i za sve $j, k \in \mathbb{N}$, označimo s $r_j^{(k)}$ sadržaj registra $\mathcal{R}_j$, a s $\text{PC}^{(k)} (< n)$ sadržaj programskog brojača, nakon k koraka izračunavanja.

Promotrimo σ-strukturu $\mathfrak{N}$, s nosačem $\mathbb{N}$ i interpretacijom zadanom s: $o^{\mathfrak{N}} := 0$; $s^{\mathfrak{N}} := S$c; te za sve $i \le n$,

$$R_i^{\mathfrak{N}} := \Big\{\vec{c} \in \mathbb{N}^m \;\Big|\; \exists k \Big(\text{PC}^{(k)} = i \wedge (\forall j < m)\big(r_j^{(k)} = c_{j+1}\big)\Big)\Big\}.$$

U toj strukturi vrijedi π_t i sve ι_i, pa mora vrijediti i ζ.

No $\mathfrak{N} \vDash \zeta$ znači $R_n^{\mathfrak{N}} \neq \varnothing$, odnosno $\exists k \big(\text{PC}^{(k)} = n \wedge \cdots\big)$, što je kontradikcija s pretpostavkom da izračunavanje ne stane.

Dokaz Churchova teorema

Korolar

Za svaki $t \in \mathbb{N}$ vrijedi $K(t) \iff (\vDash \varphi_t)$, gdje je

$$\varphi_t := \Big(\pi_t \to \big(\overline{\iota_0 \wedge \iota_1 \wedge \cdots \wedge \iota_{n-1}} \to \zeta\big)\Big).$$

Teorem (Church)

Ne postoji algoritam koji za proizvoljnu formulu prvog reda odlučuje je li valjana.

Dokaz.

Kad bi takav algoritam postojao, mogli bismo pomoću njega napisati algoritam koji prima prirodni broj t, konstruira formulu φ_t, ustanovi je li φ_t valjana, te ako jest vrati 1, a inače vrati 0. Taj algoritam bi računao χ_K, pa bi po Church–Turingovoj tezi K bila rekurzivna, kontradikcija. $\square$

Formule prvog reda kao formalni jezik

Algoritam na prethodnom slajdu opisali smo samo intuitivno, jer barata formulama umjesto prirodnih brojeva, a nismo opisali kodiranje formula. Ideje iz Turing-izračunljivosti mogu nam pomoći da i to formaliziramo: svaka se formula logike prvog reda može zapisati kao riječ nad abecedom

$$\Sigma_{\log} := \{\mathtt{x}, \mathtt{c}, \mathtt{f}, \mathtt{R}, \mathtt{,}, \mathtt{'}, \mathtt{(}, \mathtt{)}, \neg, \to, \forall, \#\},$$
$$\text{kōd:} \quad 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ A\ B\ C$$

pri čemu se svi logički veznici i kvantifikatori mogu izraziti pomoću navedenih (detalji u: Vuković, *Matematička logika*).

Indeksiranje varijabli i nelogičkih simbola piše se pomoću ponavljanja crtice ': recimo, x_0 se piše kao x, a f_3 kao f'''. Mjesnost se može zaključiti brojeći zareze , između zagradā (). Jednakost shvaćamo kao R_0^2, dakle $c_0 = x_1$ pišemo kao R(c,x').

Simbol # se ne pojavljuje u zapisu pojedine formule, nego služi kao separator kod pisanja konačnih nizova formula, npr. dokaza.

Kodiranje formula prvog reda

Sada je jasno da svaku formulu, čak svaki konačni niz formula, možemo shvatiti kao zapis broja u pomaknutoj bazi $b' = 12$ ($\mathbb{N}\Sigma_{\log}$ je zadano redoslijedom kojim su simboli navedeni na prethodnom slajdu; npr. $\langle R \rangle = 4$, $\langle c_2 \rangle = \langle \mathtt{c''} \rangle = (266)_{12} = 366$).

Nije jednostavno, ali može se pokazati da je to zaista kodiranje: skup kodova formulā prvog reda (jednomjesna relacija $\langle F1 \rangle$) je rekurzivan, te se osnovne metode kao što su introdukcija ili eliminacija veznika i kvantifikatora, pravila zaključivanja, supstitucija terma za varijablu i slično, sve mogu realizirati kao primitivno rekurzivne funkcije na kodovima.

Operacija $\widehat{12}$ je pritom vrlo korisna.
Za detalje: Smullyan, *Gödel's Incompleteness Theorems*.

Gradivo Interpretacije programa je korisno

Primjer

Beskontekstna gramatika (tipovi tokena su u zadnjem redu)

$$S \to P\,(\,U\,) \mid \neg S \mid (\,S \to S\,) \mid \forall V S$$
$$T \to V \mid C \mid F(U) \qquad U \to T \mid U\,,T$$
$$V \to x \mid V' \qquad C \to c \mid C' \qquad F \to f \mid F' \qquad P \to R \mid P'$$

generira jezik F1 nad abecedom $\Sigma_{\log}$.

Iz toga slijedi da je taj jezik beskontekstan, pa je rekurzivan.

Zadatak

Je li svaki beskontekstni jezik *primitivno* rekurzivan?

Kodirana izreka Churchova teorema

Zadatak

- Napišite formulu φ_t u ekvivalentnom obliku $\hat{\varphi}_t \in F1$.
- Dokažite da je funkcija Phi^1, zadana s $\mathsf{Phi}(t) := \langle \hat{\varphi}_t \rangle$, primitivno rekurzivna.

Korolar

Jezik $\mathit{Valid} \subset F1$ svih valjanih formula nije rekurzivan.

Dokaz.

Samo treba primijetiti da je $t \in K \Leftrightarrow \mathsf{Phi}(t) \in \langle \mathit{Valid} \rangle$, odnosno $\chi_K = \chi_{\langle \mathit{Valid} \rangle} \circ \mathsf{Phi}$. Kad bi Valid bio rekurzivan, tada bi χ_K bila rekurzivna kao kompozicija rekurzivnih, kontradikcija. □

Prema Gödelovu prvom teoremu nepotpunosti

Pogledajmo sada skup rečenicā $\mathcal{G}\ddot{o} := \{\neg\varphi_t \mid t \in K^c\}$.
Prema lemi koju smo dokazali prije Churchova teorema, ako $\mathfrak{N} \models \varphi_t$, tada P-izračunavanje s t stane, pa je $t \in K$.

Kontrapozicijom, za sve $t \in K^c$ je $\mathfrak{N} \not\models \varphi_t$, a kako je φ_t rečenica, zapravo vrijedi $\mathfrak{N} \models \neg\varphi_t$. To znači $(\mathbb{N}, 0, \mathsf{Sc}, R_0^{\mathfrak{N}}, \ldots, R_n^{\mathfrak{N}}) \models \mathcal{G}\ddot{o}$, odnosno sve rečenice iz $\mathcal{G}\ddot{o}$ vrijede u standardnom modelu prirodnih brojeva (obogaćenom s nekim relacijama R_i, ali one se barem načelno mogu svesti na zbrajanje i množenje).

Kada bi neka dovoljno lijepa aksiomatika te strukture (recimo, PA) mogla dokazati svaku formulu iz $\mathcal{G}\ddot{o}$, tada bismo opet imali svojevrstan algoritam za odluku Kleenejeva skupa: *paralelno* pustimo P da računa $\mathsf{Russell}(t)$ i enumerator nad $\Sigma_{\log}$ da generira dokaze u PA, za svaki od kojih provjerimo završava li s $\neg\hat{\varphi}_t$. Kontradikcija s Church–Turingovom tezom znači da postoji *Gödelova rečenica*: istinita na $\mathbb{N}$, ali nedokaziva u PA.

Funkcije višeg reda

Rekli smo da nam funkcijski interpreter za RAM-programe pruža mogućnost metaprogramiranja, odnosno prijenosa (parcijalno rekurzivnih) funkcija kao argumenata, i vraćanja istih iz, funkcija „višeg reda" koje pišemo.

Jednostavan primjer smo vidjeli u teoremu o grananju (parcijalno rekurzivna verzija), gdje smo selekcijom ispravnog indeksa zaobišli nedefiniranost irelevantnih redaka.

Tamo uopće nismo trebali mijenjati indekse, već samo odabrati pravi — ali općenito možemo zamisliti razne transformacije na RAM-programima kao izračunljive (npr. primitivno rekurzivne) funkcije na indeksima. Recimo, kad smo dokazivali zatvorenost skupa RAM-izračunljivih funkcija na razne operatore ($\circ$, PR, μ), mogli bismo svaku od tih konstrukcija shvatiti kao preslikavanje indeksa, npr. $\mathsf{primRecurse}_k(g, h)$ koje paru (indeks za G^k, indeks za H^{k+2}) pridruži indeks za $(G \,\mathsf{PR}\, H)^{k+1}$.

Specijalizacija funkcije

Jednu jednostavnu transformaciju funkcije napravit ćemo eksplicitno kao primitivno rekurzivnu funkciju na indeksima, a pomoću nje ćemo napraviti sve ostalo što nam treba.

U procesu ćemo dobiti i *teorem rekurzije*, koji nam daje mogućnost pisanja i rješavanja općih rekurzija, ne izlazeći iz okvira parcijalno rekurzivnih / RAM-izračunljivih funkcija.

Definicija

Neka su $k, l \in \mathbb{N}_+$, g^{k+l} funkcija i $\vec{y} \in \mathbb{N}^l$. Za funkciju f^k zadanu s $f(\vec{x}) := g(\vec{x}, \vec{y})$ kažemo da je dobivena iz g *specijalizacijom* zadnjih l argumenata na $\vec{y}$. Pišemo $f = \$(\vec{y}, g)$.

Primjer: $\{e\}^k = \$(e, \mathsf{comp}_k)$. Nađite još primjera u kolegiju!

Teorem o parametru

Teorem

Za svaki $k \in \mathbb{N}_+$, postoji primitivno rekurzivna funkcija S_k^2, takva da je za sve $(y, e) \in \mathbb{N}^2$, $S_k(y, e) \in \mathsf{Prog}$ indeks funkcije dobivene iz $\{e\}^{k+1}$ specijalizacijom zadnjeg argumenta na y.

Zadatak: Dokažite teorem o parametru kroz ove korake:

Funkcija dBase preslikava tip instrukcije u prim-broj koji se potencira odredištem u kodu instrukcije (1 ako takvog nema).

$\mathsf{Shift}(y, \ulcorner I \urcorner)$ pomiče odredište (ako postoji) instrukcije I za y.

Napišite makro Y^* semantike $r_{k+1} = 0 \Rightarrow r'_{k+1} = y$.

Napišite makro-program Q koji računa $\$(y, \{\ulcorner P \urcorner\}^{k+1})$.

Kōd spljoštenja $\ulcorner Q^\flat \urcorner$ prikažite kao konkatenaciju dvije funkcije (dobivene pomoću operatora povijesti). Definirajte te funkcije.

Ne zaboravite smisleno definirati i $S_k(y, e) \in \mathsf{Prog}$ za $e \in \mathsf{Prog}^c$!

Teorem o parametrima

Iteriranjem upravo definirane funkcije S_k^2 možemo specijalizirati više zadnjih argumenata odjednom.

Korolar

Za sve $k, l \in \mathbb{N}_+$ postoji primitivno rekurzivna funkcija S_k^{l+1} takva da je za sve $(\vec{y}, e) \in \mathbb{N}^{l+1}$, $S_k(\vec{y}, e) \in \mathsf{Prog}$ indeks funkcije dobivene iz $\{e\}^{k+1}$ specijalizacijom zadnjih l argumenata na $\vec{y}$.

Dokaz.

Indukcijom po l (za svaki k pojedinačno). Raspišite! □

Primjer: $S_3(7, 2, 9, 4, e) = S_3\big(7, S_4\big(2, S_5(9, S_6(4, e))\big)\big)$ ZZE2

Primitivna rekurzivnost komponiranja

Propozicija

Za sve $k, l \in \mathbb{N}_+$ postoji primitivno rekurzivna funkcija $\mathsf{compose}_k^{l+1}$ takva da je za sve $(\vec{g}, h) \in \mathbb{N}^{l+1}$,

$$\{\mathsf{compose}_k(\vec{g}^l, h)\}^k = \{h\}^l \circ \big(\{g_1\}^k, \ldots, \{g_l\}^k\big).$$

Dokaz.

$$F_{k,l}(\vec{x}^k, \vec{g}^l, h) :\simeq \mathsf{comp}_l\big(\mathsf{comp}_k(\vec{x}, g_1), \ldots, \mathsf{comp}_k(\vec{x}, g_l), h\big)$$

je parcijalno rekurzivna; označimo s $e_{k,l}$ jedan njen indeks. Specijalizacijom zadnjih $l+1$ argumenata dobivamo:

$$\mathsf{compose}_k(\vec{g}^l, h) := S_k(\vec{g}^l, h, e_{k,l})$$
$$[\,\mathsf{compose}_k^{l+1} := \$(e_{k,l}, S_k^{l+2})\,].$$ □

Rekurzivne definicije

Definicija

Neka je $k \in \mathbb{N}_+$ i G^{k+1} parcijalno rekurzivna funkcija. Jednadžbu

$$\mathsf{comp}_k(\vec{x}, e) \simeq G(\vec{x}, e)$$

(s nepoznanicom e, za sve $\vec{x}$) zovemo *općom rekurzijom*. Ako e zadovoljava tu jednadžbu, $\{e\}^k$ zovemo *rješenjem* te rekurzije.

Tri su zanimljiva pitanja vezana uz opće rekurzije:

1. imaju li uopće (parcijalno rekurzivno) rješenje;
2. je li ono jedinstveno (kao funkcija, ne kao indeks!);
3. je li to jedinstveno rješenje totalno (rekurzivno).

Odgovori na drugo i treće pitanje ovise o G.
[Npr. ako je G rekurzivna, svako rješenje je totalno.]
No odgovor na prvo pitanje uvijek je potvrdan.

Primjeri općih rekurzija

- $\mathsf{comp}_3(\vec{x}, e) \simeq \mathsf{comp}_3(\vec{x}, e)$:
  svaka parcijalno rekurzivna tromjesna funkcija je rješenje.
- $\mathsf{comp}_3(\vec{x}, e) \simeq \mathsf{comp}_3(\vec{x}, e) + 1$: jedinstveno rješenje je $\otimes^3$.
- $\mathsf{comp}_1(x, e) \simeq \begin{cases} 0, & x = 0 \\ \mathsf{comp}_1(1, e), & x = 1 \\ \mathsf{comp}_1(x \dot{-} 2, e) + 2, & x \geq 2 \end{cases}$:
  beskonačno mnogo rješenja, sva osim jednog su totalna.
- $\mathsf{comp}_1(x, e) \simeq \begin{cases} 0, & x < 2 \\ \mathsf{comp}_1(x \mathbin{/\!/} 2, e), & x \in \mathbb{N}_+ \wedge 2 \mid x \\ \mathsf{comp}_1(3x + 1, e), & \text{inače} \end{cases}$:
  neriješen je problem (Collatzova slutnja)
  je li rješenje jedinstveno odnosno rekurzivno (nulfunkcija).

Parcijalna jednakost u funkcijskim jednadžbama

Budući da su indeksi k-mjesnih funkcija definirani pomoću univerzalne funkcije, specifikaciju od S_k^{l+1} možemo napisati i na sljedeći „formalniji" način:

$$\mathsf{comp}_{k+l}(\vec{x}, \vec{y}, e) \simeq \mathsf{comp}_k\big(\vec{x}, S_k(\vec{y}, e)\big).$$

Upotrijebili smo znak $\simeq$ kako bismo istaknuli da izrazi s lijeve i desne strane ne moraju biti uvijek definirani — ali tada moraju oba biti nedefinirani za iste vrijednosti $\vec{x}$, $\vec{y}$ i e.

To je samo točkovni zapis simboličke funkcijske jednadžbe

$$\mathsf{comp}_{k+l} = \mathsf{comp}_k \circ \big(I_1^{k+l+1}, \ldots, I_k^{k+l+1}, S_k \circ (I_{k+1}^{k+l+1}, \ldots, I_{k+l+1}^{k+l+1})\big)$$

koja uključuje i jednakost domenā, ali je od nje puno čitljiviji.

Pokušaj daljih primjena

Primjer

Neka je a neki indeks za add^2. Tada je (za svaki k) $\mathsf{plus}_k^2 := \$(a, e_{k,3}, S_k^4)$ primitivno rekurzivna, te je $\mathsf{plus}_k(m, n) := \mathsf{compose}_k(m, n, a)$ indeks funkcije $\{m\}^k + \{n\}^k$.

Nažalost, $\mathsf{primRecurse}$ ne možemo dobiti tako jeftino: dobijemo

$$F(\vec{x}, y, g, h) \simeq \begin{cases} \mathsf{comp}_k(\vec{x}, g), & y = 0 \\ \mathsf{comp}_{k+2}\big(\vec{x}, \mathsf{pd}(y), F(\vec{x}, \mathsf{pd}(y), g, h), h\big), & \text{inače} \end{cases}$$

i sada desno imamo F koje se ne možemo tako lako riješiti.

Ipak, možemo je (odnosno njen indeks) shvatiti kao parametar!

Svaka funkcijska jednadžba poput gornje može se svesti na oblik $\mathsf{comp}_l(\vec{t}, f) \simeq L(\vec{t}, f)$ s nepoznanicom f, za neku parcijalno rekurzivnu funkciju L^{l+1}. Taj oblik zovemo *općom rekurzijom*.

Specijalne rekurzije

Primjer

Iz zapisa

$$\mathsf{comp}_{k+1}(\vec{x}, y, e) \simeq \begin{cases} G(\vec{x}), & y = 0 \\ H\big(\vec{x}, \mathsf{pd}(y), \mathsf{comp}_{k+1}(\vec{x}, \mathsf{pd}(y), e)\big), & \text{inače} \end{cases}$$

vidimo: primitivna rekurzija je specijalni slučaj opće rekurzije.
[Ako su G i H rekurzivne, rješenje je jedinstveno i rekurzivno.]

Zadatak

Prikažite

- rekurziju s poviješću,
- simultanu rekurziju

u obliku opće rekurzije!

Dijagonalna funkcija D_k

Intuitivno, rješenje opće rekurzije mora biti funkcija koja „zna" svoj indeks i može ga koristiti u svom radu (kao argument). Takve funkcije zovemo *dijagonalama*.

Definicija

Neka je $k \in \mathbb{N}_+$ te $h \in \mathbb{N}$. *Dijagonala* za $H := \{h\}^{k+1}$ je funkcija ∂H^k zadana s $\partial H(\vec{x}) :\simeq H(\vec{x}, h)$, odnosno $\partial H := \$(h, H)$.

Propozicija

Za svaki $k \in \mathbb{N}_+$ postoji primitivno rekurzivna funkcija D_k^1 takva da je za svaki $h \in \mathbb{N}$, $D_k(h)$ neki indeks dijagonale za $\{h\}^{k+1}$.

Dokaz.

$H(\vec{x}, h) \simeq \{h\}^{k+1}(\vec{x}, h) \simeq \mathsf{comp}_{k+1}(\vec{x}, h, h) \simeq \mathsf{comp}_k\big(\vec{x}, S_k(h, h)\big)$ znači da možemo definirati $D_k(h) := S_k(h, h)$. □

Teorem rekurzije

Teorem

Za svaki $k \in \mathbb{N}_+$, za svaku parcijalno rekurzivnu funkciju G^{k+1}, postoji $e \in \mathbb{N}$ takav da za sve $\vec{x} \in \mathbb{N}^k$ vrijedi $\mathsf{comp}(\vec{x}, e) \simeq G(\vec{x}, e)$.

Dokaz.

Kao što je objašnjeno na vrhu prethodnog slajda, tražimo e u obliku $e = D_k(h)$ za indeks h neke funkcije H. Dakle želimo

$$G\big(\vec{x}, D_k(h)\big) \overset{?}{\simeq} \mathsf{comp}_k\big(\vec{x}, D_k(h)\big) \simeq \mathsf{comp}_{k+1}(\vec{x}, h, h) \simeq H(\vec{x}, h),$$

što će vrijediti ako definiramo $H(\vec{x}, y) :\simeq G\big(\vec{x}, D_k(y)\big)$ za *sve* y.
Funkcija H je parcijalno rekurzivna (zašto?), pa ima indeks h.
Sada je po gornjem $e := D_k(h)$ traženi broj.
[Drugim riječima, ∂H je rješenje opće rekurzije.] □

ZZE1,3

Primjena teorema rekurzije: Ackermannova funkcija

Definiramo četveromjesnu parcijalno rekurzivnu funkciju grananjem:

$$G(x, y, z, e) :\simeq \begin{cases} y + 1, \text{ (sljedbenik)} & z = 0 \\ x, \text{ (zbrajanje s nulom)} & y = 0 \wedge z = 1 \\ 0, \text{ (množenje nulom)} & y = 0 \wedge z = 2 \\ 1, \text{ (potenciranje, \dots nulom)} & y = 0 \wedge z \geq 3 \\ \mathsf{comp}\big(x, \mathsf{comp}(x, \mathsf{pd}(y), z, e), \\ \qquad \mathsf{pd}(z), e\big), & y > 0 \wedge z > 0 \end{cases}$$

Po teoremu rekurzije, postoji $a \in \mathbb{N}$ takav da za sve $(x, y, z) \in \mathbb{N}^3$ vrijedi $\mathsf{comp}_3(x, y, z, a) \simeq G(x, y, z, a)$. Indukcijom po z i y sada se vidi da je $\{a\}^3 = A$ (dokažite!), pa A ima indeks.
Kako smo već vidjeli da je totalna, A je rekurzivna funkcija.

A^3 je jedna od brojnih varijanti *Ackermannove funkcije*.
Druga često korištena varijanta je $A^2(x, y) := A^3(2, y + 3, x) - 3$.
Svima je zajedničko računanje ugniježđenom rekurzijom.

Riceov teorem

Teorem

Neka je $\varnothing \subset S \subset \mathbb{N}$, te postoji $k \in \mathbb{N}_+$ takav da je S k-invarijantan. Tada S nije rekurzivan.

Dokaz.

Pretpostavimo suprotno: neka je $k \in \mathbb{N}_+$, i S neprazan rekurzivan k-invarijantan pravi podskup od $\mathbb{N}$.

Jer je $S \subset \mathbb{N}$, postoji $n \in S^c$. Jer je S neprazan, postoji $s \in S$.
Jer je S rekurzivan, funkcija $F^1 := \mathsf{if}\{S : C^1_n, C^1_s\}$ je rekurzivna, pa po teoremu o fiksnoj točki postoji $e \in \mathbb{N}$ za koji je $e \approx_k F(e)$.

Ako je $e \in S$, k-invarijantnost daje $n = F(e) \in S$, kontradikcija.
Ako pak $e \notin S$, tada je $F(e) = s \in S$, pa k-invarijantnost daje $e \in S$, kontradikcija. Dakle, pretpostavka je kriva. □

Svođenje (redukcija) relacija

Definicija

Neka su $k, l \in \mathbb{N}_+$, te T^k i S^l relacije. Kažemo da je T *sviđiva (reducibilna)* na S ako postoje rekurzivne funkcije $F^k_1, \dots, F^k_l$ takve da je $\chi_T = \chi_S \circ (F_1, \dots, F_k)$. Pišemo $T \preceq S$.

Zatvorenost skupa rekurzivnih funkcija na kompoziciju znači da je svaka relacija sviđiva na rekurzivnu, i sama rekurzivna.

Ako relacije gledamo kao probleme odlučivanja, $T \preceq S$ znači da je problem T „lakši" od S. To ne znači puno kad imamo samo dva stupnja težine (tehnički, možemo gledati i primitivno rekurzivne relacije kao zasebni stupanj, ali teško je naći „Ackermannovu relaciju"...), no sada ćemo dodati još jedan.

Toranj aritmetičkih operacija

Sljedbenik, zbrajanje, množenje, potenciranje, tetracija, ...
svaka od tih operacija dobivena je ponavljanjem prethodne.
Označimo ih s $A_0 = Sc \circ I^2_2$, $A_1 = \mathsf{add}^2$, $A_2 = \mathsf{mul}^2$, $A_3 = \mathsf{pow}$, ...
Svaka A_n je primitivno rekurzivna
— za $n < 4$ smo to već pokazali; zatim, lako se vidi

$$A_{n+1}(x, 0) = 1 \quad (\text{za } n > 1)$$
$$A_{n+1}(x, y + 1) = A_n\big(x, A_{n+1}(x, y)\big)$$

dakle $A_{n+1} = C^1_1 \mathbin{\mathsf{PR}} A_n \circ (I^3_1, I^3_3)$, pa dalje možemo indukcijom — no što je s funkcijom $A(x, y, z) := A_z(x, y)$? Očito je totalna jer su sve A_z totalne, i u intuitivnom smislu je izračunljiva, pa bi po Church–Turingovoj tezi trebala biti rekurzivna.
Poteškoća je u tome da nije primitivno rekurzivna (intuitivno, z kaže koliko puta moramo primijeniti operator PR u definiciji; ne može manje jer inače A ne raste dovoljno brzo), no primjena teorema rekurzije lako daje rekurzivnost od A^3.

Teorem o fiksnoj točki; k-invarijantnost

Korolar

Neka je $k \in \mathbb{N}_+$ te F^1 rekurzivna funkcija.
Tada postoji $e \in \mathbb{N}$ takav da je $\{e\}^k = \{F(e)\}^k$ [$\Longleftrightarrow$: $e \approx_k F(e)$].

Dokaz.

Primijenimo teorem rekurzije na $\mathsf{comp}_k(\vec{x}, e) \simeq \mathsf{comp}_k\big(\vec{x}, F(e)\big)$.
Tako dobijemo $e \in \mathbb{N}$, a jer je F totalna, imamo i $f := F(e) \in \mathbb{N}$, te su e i f indeksi iste k-mjesne funkcije. □

Definicija

Neka je $S \subseteq \mathbb{N}$, te $k \in \mathbb{N}_+$. Kažemo da je S k-*invarijantan* ako je zatvoren na $\approx_k$: $e \in S$ i $\{e\}^k = \{f\}^k$ povlače $f \in S$.

Zadatak

Dokažite: S je k-invarijantan ako i samo ako postoji skup funkcija $\mathcal{F}$ takav da je $S = \{e \in \mathbb{N} \mid \{e\}^k \in \mathcal{F}\} =: \ulcorner \mathcal{F} \urcorner$.

Uputstvo za upotrebu Riceova teorema

Želimo dokazati da skup S nije rekurzivan. [ZZF]

- Pomoću skupa S zapišemo skup $T \subseteq \mathbb{N}$ u kojem se nalaze indeksi nekih funkcija fiksne mjesnosti k. (Skup svih takvih funkcija označimo s $\mathcal{F}$.)
- Nađemo neki indeks koji jest u T (tako da nađemo neku jednostavnu funkciju u $\mathcal{F}$, napišemo RAM-program koji je računa, i kodiramo ga). [Korisno: $\{0\}^k = \otimes^k$, $\{1\}^k = C^k_0$.]
- Analognim postupkom nađemo neki indeks koji nije u T.
- Dokažemo da je T k-invarijantan, gdje je k mjesnost koju smo fiksirali u prvom koraku. Ovo je najčešće jednostavno: ako je $\{e\}^k = \{f\}^k$ i $\{e\}^k \in \mathcal{F}$, tada je i $\{f\}^k \in \mathcal{F}$.
- Dokažemo da iz pretpostavke da je S rekurzivan slijedi rekurzivnost od T [napišemo χ_T, najčešće kompozicijom, pomoću χ_S (*svedemo* T na S): detalji na sljedećem slajdu], što je kontradikcija s Riceovim teoremom.

Poluodlučivi problemi i njihova formalizacija

Znamo da Kleenejev skup nije rekurzivan: ne možemo za svaki prirodni broj t u konačno mnogo koraka odgovoriti je li $t \in K$.
Ipak, ako je odgovor „da", to ćemo sigurno saznati čim računanje $\mathsf{Russell}(t)$ završi.
Jedino ako je odgovor „ne", *možda* to nećemo nikada saznati (možemo ponekad, recimo kad $t \notin \mathsf{Prog}$, ali ne uvijek).

Takvi problemi su *poluodlučivi*: „odlučivi" su u slučaju pozitivnog odgovora. Zbog univerzalnosti RAM-modela, *halting problem* je univerzalni primjer poluodlučivog problema, što nas motivira da poluodlučive probleme općenito modeliramo kao domene izračunljivih funkcija.

Definicija

Neka je $k \in \mathbb{N}_+$. Kažemo da je relacija R^k *rekurzivno prebrojiva* ako postoji parcijalno rekurzivna funkcija f^k takva da je $R = \mathcal{D}_f$.

Svođenje čuva rekurzivnu prebrojivost

Propozicija

Svaka relacija svediva na rekurzivno prebrojivu relaciju je i sama rekurzivno prebrojiva.

Dokaz.

Neka su $k, l \in \mathbb{N}_+$ te $T^k \preceq S^l = \mathcal{D}_{f^l}$. Tada postoje rekurzivne funkcije $F_1^k, \ldots, F_l^k$ takve da je $\chi_T = \chi_S \circ (F_1, \ldots, F_l)$.
Označimo $g^k := f^l \circ (F_1, \ldots, F_l)$. Kako su sve F_i rekurzivne, i time totalne, formula za domenu kompozicije daje

$$\begin{aligned} \vec{x}^k \in \mathcal{D}_g &\iff \vec{x} \in \textstyle\bigcap_{i=1}^l \mathbb{N}^k \wedge \big(F_i(\vec{x})\big)_{i=1}^l \in \mathcal{D}_f = S \\ &\iff 1 = \chi_S\big(F_i(\vec{x})\big)_{i=1}^l = \chi_T(\vec{x}) \iff \vec{x} \in T \end{aligned}$$

pa je $T = \mathcal{D}_g$ domena parcijalno rekurzivne funkcije g, dakle rekurzivno prebrojiva. □

Restrikcija na rekurzivno prebrojiv skup

Lema (o restrikciji)

Restrikcija parcijalno rekurzivne funkcije na rekurzivno prebrojiv skup je parcijalno rekurzivna.

Dokaz.

Ako su mjesnosti različite, restrikcija je prazna, pa je parcijalno rekurzivna kao minimizacija prazne relacije.
Inače, označimo zajedničku mjesnost s k, i neka su F^k i G^k parcijalno rekurzivne te $S^k = \mathcal{D}_{G^k}$.
Tada je $F|_S = I_1^2 \circ (F, G)$. □

Raspišite detalje u gornjem dokazu!

Projekcijska karakterizacija rekurzivne prebrojivosti

Teorem

Neka je $k \in \mathbb{N}_+$ te R^k relacija. Tada je R rekurzivno prebrojiva ako i samo ako je $R = \exists_* P$ za neku (primitivno) rekurzivnu P^{k+1}.

Dokaz.

($\Rightarrow$) Neka je $R = \mathcal{D}_{f^k}$ te e neki indeks za f. Po Kleenejevu teoremu o normalnoj formi, $R(\vec{x}) \Leftrightarrow \exists y T_k(\vec{x}, e, y)$, a relacija $P(\vec{x}, y) :\Leftrightarrow T_k(\vec{x}, e, y)$ je primitivno rekurzivna.

($\Leftarrow$) Neka je $R = \exists_* P$, gdje je P (primitivno) rekurzivna. Tada je $\mathcal{D}_{\mu P} = \exists_* P = R$, a μP je parcijalno rekurzivna kao minimizacija (primitivno) rekurzivne relacije. □

Možemo proučavati djelovanje raznih kvantifikatora ($\exists_*, \forall_*$) na rekurzivne relacije. U tom kontekstu (*aritmetička hijerarhija*) projekcije se zovu Σ_1^0. Detaljnije u: Vuković, *Izračunljivost*.

Kodiranje parova brojeva

Skup $\mathbb{N}^2 \subset \mathbb{N}^*$ se svakako može kodirati (funkcijom Code^2), ali za osnovne metode dobro je uvesti posebne oznake.

$$\mathrm{fst}(p) := p[0] \qquad \mathrm{snd}(p) := p[1]$$

Definicija

Neka je $k \in \mathbb{N}_+$ te R^{k+1} relacija. Za relaciju $\hat{R}^k$ definiranu s

$$\hat{R}(\vec{x}, y) :\iff R\big(\vec{x}, \mathrm{fst}(y), \mathrm{snd}(y)\big)$$

kažemo da je dobivena iz R *kontrakcijom* zadnja 2 argumenta.

Funkcije $\mathrm{fst} = \$(0, \mathrm{part})$ i $\mathrm{snd} = \$(1, \mathrm{part})$ su (primitivno) rekurzivne, pa je $\hat{R} \preceq R$.
Također, lako vidimo $R(\vec{x}, y, z) \Leftrightarrow \hat{R}(\vec{x}, \langle y, z\rangle)$, iz čega $R \preceq \hat{R}$.

Rekurzivnost je strogo jača od rekurzivne prebrojivosti

Propozicija

Svaka rekurzivna relacija je rekurzivno prebrojiva.

Dokaz.

Neka je $k \in \mathbb{N}_+$ te R^k rekurzivna. Po teoremu o grananju (parcijalno rekurzivna verzija), $f := \mathrm{if}\{R^k : C_0^k\}$ je parcijalno rekurzivna funkcija, s domenom $\mathcal{D}_{C_0^k} \cap R^k = \mathbb{N}^k \cap R^k = R^k$. □

Zadatak

Dokažite: $f(\vec{x}) :\simeq \mu y\big(\mathrm{Sc}(y) = \chi_R(\vec{x})\big)$ je točkovna definicija od f.

Napomena

Obrat ne vrijedi: $K := \mathcal{D}_{\mathrm{Russell}}$ je rekurzivno prebrojiv po definiciji, ali nije rekurzivan.

Presjek konačno mnogo rekurzivno prebrojivih relacija

Propozicija

Presjek dvije rekurzivno prebrojive relacije je rekurzivno prebrojiva relacija.

Dokaz.

Neka su $S = \mathcal{D}_f$ i T rekurzivno prebrojive relacije.
Tada je $S \cap T = \mathcal{D}_f \cap T = \mathcal{D}_{f|_T}$, što je domena parcijalno rekurzivne funkcije po lemi o restrikciji. □

Korolar

Presjek konačno mnogo rekurzivno prebrojivih relacija također je rekurzivno prebrojiva relacija.

Vrijedi i zatvorenost na (čak rekurzivno prebrojive) unije, ali za to nam trebaju moćniji alati — poput projekcije $\exists_*$.

Zatvorenost na konačne unije

Propozicija

Neka su $k, l \in \mathbb{N}_+$ te $R_1^k, \ldots, R_l^k$ rekurzivno prebrojive relacije.
Tada je i njihova unija $\bigcup_{i=1}^l R_i$ rekurzivno prebrojiva relacija.

Dokaz.

Za svaki $i \in [1 \cdot\cdot\, l]$ po projekcijskoj karakterizaciji postoji rekurzivna relacija P_i^{k+1} takva da je $R_i = \exists_* P_i$. Sada vrijedi

$$\begin{aligned} \vec{x} \in \bigcup_{i=1}^l R_i &\iff (\exists i \in [1 \cdot\cdot\, l])(\exists y) P_i(\vec{x}, y) \iff \\ &\iff (\exists y)(\exists i \in [1 \cdot\cdot\, l]) P_i(\vec{x}, y) \iff \exists y\Big((\vec{x}, y) \in \bigcup_{i=1}^l P_i\Big), \end{aligned}$$

pa je $\bigcup_{i=1}^l R_i = \exists_*\big(\bigcup_{i=1}^l P_i\big)$ projekcija rekurzivne relacije. □

Gore smo imali ograničenu (po i) i neograničenu (po y) egzistencijalnu kvantifikaciju. Što bismo s dvije neograničene?

Kontrakcija egzistencijalnih kvantifikatora

Lema

Za svaku relaciju R mjesnosti barem 3 vrijedi $\exists_* \exists_* R = \exists_* \hat{R}$.

Dokaz.

Neka je $k \in \mathbb{N}_+$, neka je R^{k+2} (dakle $\hat{R}^{k+1}$) relacija, i neka je $\vec{x} \in \mathbb{N}^k$ proizvoljan.

($\subseteq$) Ako je $\vec{x} \in \exists_* \exists_* R$, to znači da postoji y takav da je $(\vec{x}, y) \in \exists_* R$, pa postoji i z takav da vrijedi $R(\vec{x}, y, z)$. Tada postoji $t := \langle y, z\rangle$ takav da vrijedi $\hat{R}(\vec{x}, t)$, odnosno $\vec{x} \in \exists_* \hat{R}$.

($\supseteq$) Ako je $\vec{x} \in \exists_* \hat{R}$, to znači da postoji $t \in \mathbb{N}$ takav da vrijedi $\hat{R}(\vec{x}, t)$, odnosno $R\big(\vec{x}, \mathrm{fst}(t), \mathrm{snd}(t)\big)$. To pak znači da postoje $y := t[0]$ i $z := t[1]$ takvi da vrijedi $R(\vec{x}, y, z)$, odnosno $(\vec{x}, y) \in \exists_* R$, pa onda i $\vec{x} \in \exists_* \exists_* R$. □

Zatvorenost na projekciju

Propozicija

Neka je R rekurzivno prebrojiva relacija mjesnosti barem 2. Tada je i njena projekcija $S := \exists_* R$ rekurzivno prebrojiva.

Dokaz.

Uvjet na mjesnost služi kako bi projekcija bila definirana: označimo njenu mjesnost s $k \in \mathbb{N}_+$. Tada je R mjesnosti $k+1$. Projekcijska karakterizacija kaže da je $R = \exists_* P$ za neku rekurzivnu relaciju P^{k+2}. Iz $\hat{P} \preceq P$ slijedi da je $\hat{P}$ također rekurzivna, te je po prethodnoj lemi $S = \exists_* R = \exists_* \exists_* P = \exists_* \hat{P}$, što je rekurzivno prebrojivo po projekcijskoj karakterizaciji. □

Korolar

Neka je $k \in \mathbb{N}_+$ te R^{k+1} rekurzivno prebrojiva relacija. Tada je i relacija zadana s $Q(y) :\Leftrightarrow \exists \vec{x}^k R(\vec{x}, y)$, rekurzivno prebrojiva.

Teorem o grafu za totalne funkcije

Teorem

Neka je $k \in \mathbb{N}_+$ te F^k totalna funkcija.
Tada je F rekurzivna ako i samo ako joj je graf $\mathcal{G}_F$ rekurzivan.

Dokaz.

($\Rightarrow$) Jer je F totalna, vrijedi $\mathcal{G}_F(\vec{x}, y) \Leftrightarrow F(\vec{x}) = y$, dakle $\mathcal{G}_F \preceq (=)$ $[\chi_{\mathcal{G}_F}(\vec{x}, y) = \chi_=(F(\vec{x}), y)]$, a jednakost je rekurzivna relacija.

($\Leftarrow$) $F = \mu \mathcal{G}_F$ je parcijalno rekurzivna kao minimizacija rekurzivne relacije, pa je rekurzivna zbog totalnosti. □

Napomena

Ovdje ne možemo staviti „primitivno" u zagrade!
$\mathcal{G}^3_{A^2}$ je primitivno rekurzivan, ali A^2 nije — Vuković za detalje.

Selektor (funkcija izbora)

Definicija

Neka je $k \in \mathbb{N}_+$, te R^{k+1} relacija i f^k funkcija.
Kažemo da je f *selektor* za R ako vrijedi $\mathcal{D}_f = \exists_* R$ i $\mathcal{G}_f \subseteq R$.

Općenito, naravno, selektor nije jedinstven — ali za grafove jest.

Propozicija

Neka je $k \in \mathbb{N}_+$, te f^k funkcija. Jedini selektor za $\mathcal{G}_f$ je upravo f.

Dokaz.

($\exists$) Znamo $\mathcal{D}_f = \exists_* \mathcal{G}_f$, a očito je $\mathcal{G}_f \subseteq \mathcal{G}_f$, pa je f selektor za $\mathcal{G}_f$.

(!) Neka je g bilo koji selektor za $\mathcal{G}_f$. Tada $\mathcal{D}_g = \exists_* \mathcal{G}_f = \mathcal{D}_f$ te za svaki $\vec{x}$ iz tog skupa vrijedi $(\vec{x}, g(\vec{x})) \in \mathcal{G}_g \subseteq \mathcal{G}_f$, što po definiciji grafa znači $g(\vec{x}) \simeq f(\vec{x})$. Dakle $g = f$. □

Primjena selektora: univerzalnost

Relacija $M(x, e, y, n) :\Leftrightarrow \mathsf{Step}(x, e, n) \wedge \mathsf{result}(\mathsf{Reg}(x, e, n)) = y$ je primitivno rekurzivna kao presjek dvije takve.
Njena (rekurzivno prebrojiva) projekcija $\exists_* M$ je graf *univerzalne funkcije* $\mathsf{univ}(\langle \vec{x} \rangle, e) :\simeq \{e\}^k(\vec{x})$. $\mathsf{univ} = \mathsf{fst} \circ \mu \hat{M}$ je parcijalno rekurzivna kao (jedini) selektor za vlastiti graf.
Domena joj je *halting problem*, $\mathrm{HALT} := \mathcal{D}_{\mathsf{univ}} = \exists_* \exists_* M$.
Njenim kompiliranjem dobijemo program P_U za *univerzalni RAM-stroj* $\mathcal{S}_U$, koji u registru $\mathcal{R}_2$ dobije RAM-program P, a u $\mathcal{R}_1$ ulaz $\vec{x}$ za njega, te u $\mathcal{R}_0$ vraća rezultat P-izračunavanja s $\vec{x}$, ako i samo ako to izračunavanje stane (inače ni P_U ne stane).
Transpiliranjem P_U dobijemo *univerzalni Turingov stroj* $\mathcal{T}_U$, koji računa βuniv: na traci dobije $\bullet^{\langle w \rangle}/\bullet^{\langle P \rangle}$ (gdje P računa $\mathbb{N}\varphi$), i vraća $\bullet^{\langle \varphi(w) \rangle}$, ako i samo ako je $w \in \mathcal{D}_\varphi$. Za fiksnu Σ, lako možemo postići (obrćući traku te „našarafljujući" drugi i četvrti fragment transpiliranja) da w prima u nativnom formatu Σ^*, a *parsiranjem* (IP; teže) može tako primati i $\mathcal{T}$.

Grafovi

S grafovima smo se već upoznali kod relacijā Step i T_k.

Definicija

Neka je $k \in \mathbb{N}_+$ i f^k funkcija. *Graf* od f je relacija $\mathcal{G}^{k+1}_f$, zadana s

$$\mathcal{G}_f(\vec{x}, y) :\Longleftrightarrow \mathcal{D}_f(\vec{x}) \wedge f(\vec{x}) \simeq y.$$

[Za totalnu f, ta konjunkcija je ekvivalentna s $f(\vec{x}) = y$.]

Kažemo da relacija R^{k+1} ima *funkcijsko svojstvo* ako $R(\vec{x}, y_1)$ i $R(\vec{x}, y_2)$ povlače $y_1 = y_2$.

Otprije znamo („vertikalni test") da je relacija graf neke funkcije ako i samo ako ima funkcijsko svojstvo.

Lema

Za svaku brojevnu funkciju f vrijedi $\exists_* \mathcal{G}_f = \mathcal{D}_f$ i $\mu \mathcal{G}_f = f$.

Selekcija — intuitivno

Ključni korak u dokazu prethodnog teorema bila je primjena minimizacije, što smo mogli jer je graf bio rekurzivan.

Općenito, minimizacija poluizračunljivog skupa S neće biti izračunljiva (razmislite!), ali u specijalnom slučaju, kad S ima funkcijsko svojstvo, zapravo *hoće*.

To ćemo dokazati tako da dokažemo slabiju tvrdnju pod općenitijim uvjetom: dopuštamo da ima *više* y za koje vrijedi $S(\vec{x}, y)$, ali ne želimo najmanji, nego jednostavno *neki* takav y.

Tada, ako je y *jedinstven*, dobit ćemo upravo njega.

Tu „operaciju" (pod navodnicima jer nije deterministična) zovemo *selekcija*, a (bilo koju) funkciju f koja je obavlja (preslikava $\vec{x}$ u neki y takav da je $S(\vec{x}, y)$) zovemo *selektor*.

Naravno, kao ni minimizacija, ni selekcija nije definirana za one $\vec{x}$ koji nisu u projekciji (za koje takav y ne postoji).

Teorem o selektoru

Lema

Za svaku rekurzivno prebrojivu relaciju mjesnosti barem 2 postoji parcijalno rekurzivni selektor.

Dokaz.

Neka je $k \in \mathbb{N}_+$, te R^{k+1} rekurzivno prebrojiva relacija $(k+1 \geq 2)$. Po projekcijskoj karakterizaciji, $R = \exists_* P$ za neku rekurzivnu relaciju P^{k+2}. Iz $\hat{P} \preceq P$ slijedi da je i $\hat{P}$ rekurzivna, pa je funkcija $f := \mathsf{fst} \circ \mu \hat{P}$ parcijalno rekurzivna. Tvrdimo da je f selektor za R.

- $\exists_* R = \exists_* \exists_* P = \exists_* \hat{P} = \mathcal{D}_{\mu \hat{P}} = \mathcal{D}_{\mathsf{fst} \circ \mu \hat{P}} = \mathcal{D}_f$.
- $\mathcal{G}_f(\vec{x}, y)$ znači $\mathcal{D}_f(\vec{x})$ i $y = f(\vec{x}) = \mathsf{fst}\big(\mu t\, \hat{P}(\vec{x}, t)\big)$, dakle postoji [najmanji] t takav da vrijedi $\hat{P}(\vec{x}, t)$, te je $y = \mathsf{fst}(t)$. Uvrstivši to u definiciju kontrakcije, imamo $P\big(\vec{x}, y, \mathsf{snd}(t)\big)$, pa je $(\vec{x}, y) \in \exists_* P = R$. Dakle $\mathcal{G}_f \subseteq R$. □

Višedretvenost

Teorem o selektoru ima još jednu zanimljivu interpretaciju.

Ako zamislimo da su nam (dvije) neograničene kvantifikacije nad P ugniježđene beskonačne petlje (po y i z), kontrakcija kvantifikatora zapravo kaže da ih možemo „ispreplesti": izvršavati tako da svaki par (y, z) dođe na red. Ukratko, imamo „izračunljivi dokaz" da je $\mathbb{N}^2$ prebrojiv.

Ali to nije sve. Ako detaljnije pogledamo dokaz projekcijske karakterizacije, vidjet ćemo da petlja po z traži upravo kōd izračunavanja čija vrijednost U je $\chi_R(\vec{x}, y)$: zbog $\exists_* \mathsf{Step} = \exists_* \overset{\bullet}{T}$ možemo zamisliti da računa inkrementalno korak po korak tog izračunavanja, dok ne dobije 1 (ili zauvijek ako je rezultat 0).

Štoviše, zbog prvog odlomka, ti koraci su isprepleteni s istim takvim koracima za neke druge y.
Zaključujemo da selektor paralelno računa $\chi_R(\vec{x}, y)$ za sve y, prekidajući rad kad/ako nađe jedan y takav da je rezultat 1.

Teorem o grafu za parcijalne funkcije

Sada napokon možemo dokazati „pravi” teorem o grafu.

Teorem

Neka je $k \in \mathbb{N}_+$ te f^k funkcija. Tada je f parcijalno rekurzivna ako i samo ako joj je graf $\mathcal{G}_f$ rekurzivno prebrojiv.

Dokaz.

($\Rightarrow$) Označimo s e neki indeks za f^k. Kleenejev teorem kaže:

$$\mathcal{G}_f(\vec{x}, y) \iff \exists z \left(T_k(\vec{x}, e, z) \wedge U(z) = y\right),$$

pa tvrdnja slijedi po projekcijskoj karakterizaciji.

($\Leftarrow$) Po teoremu o selektoru $\mathcal{G}_f$ ima parcijalno rekurzivni selektor. No dokazali smo da je jedini selektor za $\mathcal{G}_f$ upravo f, pa to znači da je f parcijalno rekurzivna. □

Editiranje (konačna promjena) parcijalnih funkcija

Zadatak

Dokažite: skup rekurzivno prebrojivih relacija zatvoren je na

- desne razlike s rekurzivnim relacijama (iste mjesnosti), i
- simetrične razlike s konačnim relacijama (iste mjesnosti).

Propozicija

Neka je $k \in \mathbb{N}_+$, neka je G^k parcijalno rekurzivna funkcija, te F^k funkcija dobivena iz G konačnim brojem promjena vrijednosti, uklanjanja ili dodavanja točaka u domenu (s bilo kojom vrijednošću). Tada je i F parcijalno rekurzivna.

Dokaz.

Direktno iz teorema o grafu, gornjeg zadatka, i činjenice da je *diff* $\mathcal{G}_F \triangle \mathcal{G}_G$ konačan. Raspišite! □

Postov teorem

Teorem

Neka je $k \in \mathbb{N}_+$ te R^k relacija. Tada vrijedi:
R je rekurzivna ako i samo ako su R i R^c rekurzivno prebrojive.

Dokaz.

($\Rightarrow$) Ako je R rekurzivna, dokazali smo da je i R^c rekurzivna, a svaka rekurzivna relacija je rekurzivno prebrojiva.

($\Leftarrow$) Ako su R i R^c rekurzivno prebrojive tada, po posljednjem dokazanom teoremu o grananju, $\chi_R := \text{if}\{R : C_1^k, R^c : C_0^k\}$ je parcijalno rekurzivna funkcija. Ali χ_R je uvijek totalna, pa je zapravo rekurzivna, dakle R je rekurzivna relacija. □

Korolar

Skup K^c nije rekurzivno prebrojiv. (Dokažite!)

Teorem enumeracije

Teorem

Neka je $S \subseteq \mathbb{N}$ (jednomjesna relacija). Tada je ekvivalentno:

1. S je rekurzivno prebrojiv.
2. S je slika neke parcijalno rekurzivne funkcije.
3. S je slika neke primitivno rekurzivne funkcije ili $S = \varnothing$.

Dokaz.

($1 \Rightarrow 3$) Iz $s \in S = \mathcal{D}_{\{e\}^1}$ slijedi $S = \mathcal{I}_F$ za $F(x,t) := \begin{cases} x, & T_1(x,e,t) \\ s, & \text{inače} \end{cases}$.

Naime, $s \in S \cap \mathcal{I}_F$, a za $x \neq s$ po Kleenejevu teoremu imamo $x \in S = \mathcal{D}_{\{e\}^1} \Leftrightarrow \exists t\, T_1(x,e,t) \Leftrightarrow \exists t \left(F(x,t) = x\right) \Leftrightarrow x \in \mathcal{I}_F$.

($2 \Rightarrow 1$) $\mathcal{I}_f(y) \Leftrightarrow \exists \vec{x}\, \mathcal{G}_f(\vec{x}, y)$ je projekcija rekurzivno prebrojivog $\mathcal{G}_f$.

($3 \Rightarrow 2$) Primitivna rekurzivnost povlači parcijalnu, a $\varnothing^1 = \mathcal{I}_{\otimes^1}$. □

Teorem o grananju, rekurzivno prebrojiva verzija

Teorem

Neka su $k, l \in \mathbb{N}_+$, neka su $G_1^k, \ldots, G_l^k$ parcijalno rekurzivne funkcije te $R_1^k, \ldots, R_l^k$ u parovima disjunktne rekurzivno prebrojive relacije. Tada je funkcija $F^k := \text{if}\{R_1 : G_1, \ldots, R_l : G_l\}$ također parcijalno rekurzivna.

Dokaz.

Za svaki $i \in [1 \cdot\cdot\, l]$, označimo $H_i := G_i|_{R_i}$
— po lemi o restrikciji, to su parcijalno rekurzivne funkcije.

Sada tvrdnja slijedi iz teorema o grafu, jer je (dokažite!)

$$\mathcal{G}_F = \bigcup_{i=1}^{l} \mathcal{G}_{H_i},$$

a unija l rekurzivno prebrojivih skupova je opet rekurzivno prebrojiva. □

O *default*nim slučajevima u grananju

Pogledajmo detaljnije dosadašnje verzije teorema o grananju, s obzirom na prisutnost riječi „inače” u definiciji funkcije.

Kod (primitivno) rekurzivne verzije, slučaj „inače” je obavezan, jer bez njega nemamo osiguranu totalnost definirane funkcije. Možemo nekako drugačije osigurati da R_i čine particiju od $\mathbb{N}^k$, no to samo znači da jednu od njih možemo zamijeniti s „inače”.

Kod parcijalno rekurzivne verzije, slučaj „inače” je dozvoljen: imamo dvije funkcije, F_1 i F_2, i obje su parcijalno rekurzivne. Naravno, F_2 je specijalni slučaj od F_1, gdje stavimo $G_0 := \otimes^k$.

Kod rekurzivno prebrojive verzije, slučaj „inače” je zabranjen, jer s njime bismo mogli dobiti da je $\overline{\text{Russell}} = \text{if}\{K : \text{Russell}, Z\}$ parcijalno rekurzivna, što smo vidjeli da ne vrijedi.

Općenito, skup rekurzivno prebrojivih relacija nije zatvoren na komplementiranje. Štoviše, vrijedi ...

Veza sa slikama

Teorem o grafu daje dobru vizualizaciju poluizračunljivih relacija, barem onih s funkcijskim svojstvom:
to su grafovi izračunljivih funkcija.
No ta vizualizacija je beskorisna za *jednomjesne* relacije (skupove brojeva), kakve se najčešće proučavaju.

Povijesno, rekurzivno prebrojivi skupovi bili su promatrani kao *slike* izračunljivih (jednomjesnih) funkcija — otud dolazi i naziv, jer se radi o skupovima koji se mogu rekurzivno nanizati/nabrojiti ($a_0, a_1, a_2, \ldots$, gdje je $n \mapsto a_n$ rekurzivna).

Uz problem mjesnosti (slike su jednomjesne) imamo probleme totalnosti ($\varnothing$ očito nije slika nijedne rekurzivne funkcije), i injektivnosti (konačne skupove danas ne zovemo prebrojivima).
Ipak, mnogi rezultati se mogu dokazati.

Veza s kolegijem Interpretacija programa

Analogno rekurzivnim relacijama odnosno jezicima, možemo gledati *rekurzivno prebrojive jezike*, na tri načina.

Gledanje na njih kao na domene Turing-izračunljivih funkcija prirodno vodi na ideju Turingovih prepoznavača, dok gledanje na njih kao na slike takvih funkcija prirodno vodi na ideju Turingovih enumeratora.
Na IP ste dokazali da je to dvoje ekvivalentno.

Treće, možemo definirati da je jezik $L \subseteq \Sigma^*$ rekurzivno prebrojiv ako je skup kodova $\langle L \rangle := \{\langle w \rangle \mid w \in L\}$ rekurzivno prebrojiv.
Može se pokazati da je i ta definicija ekvivalentna gornjima.

ZZG2,3 (i sve ostalo)