Parcijalno rekurzivni funkcionali

Pavlović, Andrej

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Andrej Pavlović

PARCIJALNO REKURZIVNI FUNKCIONALI

Diplomski rad

Voditelj rada:
izv.prof.dr.sc.
Mladen Vuković

Zagreb, rujan 2017

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Ovaj rad vezan je uz granu matematičke logike, računarstva, te teorije računanja koja se zove *izračunljivost*. Ključni dio izračunljivosti iz kojeg proizlazi tema ovog rada su *parcijalno rekurzivne funkcije*. Ovdje ćemo generalizirati parcijalno rekurzivne funkcije na prirodnim brojevima (budući da ćemo isključivo dozvoljavati i raditi sa skupom prirodnih brojeva). Generalizacija će u ovom slučaju značiti da osim brojevnih argumenata, dozvoljavamo da argumenti funkcija budu i funkcije na prirodnim brojevima. Takve funkcije ćemo nazvati *funkcionalima*. Proširit ćemo definiciju parcijalno rekurzivnih funkcija na *parcijalno rekurzivne funkcionale*.

Kako bi došli do definicije parcijalno rekurzivnih funkcionala, te definirali sve pojmove vezane uz parcijalno rekurzivne funkcionale, prvo moramo govoriti o parcijalno rekurzivnim funkcijama kako bi bili u mogućnosti napraviti poveznicu i generalizaciju na parcijalno rekurzivne funkcionale. Stoga u prvom poglavlju dajemo opisne definicije osnovnih pojmova, te termine i oznake koje ćemo koristiti. Zatim govorimo o *RAM-stroj* te dajemo opisnu definiciju stroja. Intuitivno dajemo pojam izračunavanja, te samim time i pojam izračunljive funkcije. Zatim ćemo reći što su to rekurzivne funkcije te definirati klasu parcijalno rekurzivnih funkcija. Dajemo skicu dokaza osnovnog teorema koji proizlazi kao rezultat o parcijalno rekurzivnim funkcijama, a to je *Kleenijev teorem o normalnoj formi* za parcijalno rekurzivne funkcije. Definiramo što je to kod, te funkcija kodiranja.

Zatim ćemo u drugom poglavlju ovog rada dati definiciju parcijano rekurzivnog funkcionala. Definirat ćemo *generalizirani RAM-stroj* i *GRAM-izračunljive funkcije*. Modifikacijom Kleenijevog teorema o normalnoj formi za parcijalno rekurzivne funkcionale i uvođenjem aritmetičkog pristupa dokazujemo teorem o normalnoj formi za parcijalno rekurzivne funkcionale. Na kraju uvodimo pojam fiksne točke nakon kojega možemo iskazati i dokazati *prvi teorem rekurzije* za parcijalno rekurzivne funkcionale.

Poglavlje 1

Parcijalno rekurzivne funkcije

Kako bi kasnije mogli govoriti o parcijalno rekurzivnim funkcionalima, u prvom poglavlju ovog rada napravit ćemo ponavljanje iz kolegija Izračunljivost te u tu svrhu koristiti i dio nastavnog materijala profesora Mladena Vukovića predviđenog za taj kolegij. Omogućit će nam generalizaciju pojmova, definicija i teorema koja će biti potrebna kada budemo govorili o parcijalno rekurzivnim funkcionalima.

1.1 Osnovni pojmovi

Dajemo opsine definicije pojmova koje ćemo razmatrati. Kasnije ćemo te pojmove strogo definirati.

Izračunavanje je proces kod kojeg iz početnih objekata skupom pravila dobivamo izlazni rezultat. Početni objekti su unaprijed definirani i nazivamo ih **ulazni podaci**. Skup pravila ćemo zvati **algoritam** ili **program**. Kranje rezultate ćemo zvati **izlazni podaci**.

Pretpostavljamo da postoji najviše jedan izlazni podatak. Ako pak želimo promatrati izračunavanje s k izlaznih podataka, tada možemo ustvari promatrati k izračunavanja, od koje svako ima točno jedan izlazni rezultat. S druge strane, dozvoljavamo svaki **konačan** broj ulaznih podataka (uključujući i nula ulaznih podataka).

Pretpostavljamo da je za svaki algoritam ili program fiksirana **mjesnost** (broj) ulaznih podataka.

Nadalje, ne zahtjevamo da za sve ulazne podatke svaki algoritam daje izlazni rezultat. To znači da algoritam za neke ulazne podatke može računati, ali može **nikada ne stati**. Kod algoritma moramo točno znati što je akcija, odnosno **instrukcija** koja se odvija u svakom koraku izvođenja algoritma.

U uvodu smo rekli da ćemo promatrat funkcije nad prirodnim brojevima. To su funkcije oblika $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$, gdje je $\mathbb{N}$ skup prirodnih brojeva $\{0,1,2,\dots\}$, te $k \in \mathbb{N} \setminus \{0\}$.

Neka je f gore definirana funkcija, te označimo sa A proizvoljan algoritam. Smatramo

da algoritam A s k ulaznih podataka **izračunava** funkciju f ako vrijedi:

Prirodni brojevi $x_1, \ldots, x_k$ su u domeni funkcije f ako i samo ako algoritam A prilikom izračunavanja s ulaznim podacima $x_1, \ldots, x_k$ stane, te je izlazni rezultat u tom slučaju jednak $f(x_1, \ldots, x_k)$.

Smatramo da je neka funkcija $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$ **izračunljiva** ako postoji neki algoritam koji je izračunava.

1.2 Oznake

Sada navodimo osnovne termine i oznake koje ćemo dalje koristiti.

Rekli smo da će nam $\mathbb{N}$ označavati skup prirodnih brojeva {0,1,2,...}, te da promatramo samo algoritme čiji su ulazni i izlazni podaci prirodni brojevi. Nadalje, pod "skup" mislimo na neki podskup skupa $\mathbb{N}^k$, za neki $k \in \mathbb{N}$ ($k > 0$). Uređenu k-torku prirodnih brojeva $(x_1, \ldots, x_k)$ zapisujemo kraće kao $\vec{x}$. Kada kažemo "k-mjesna funkcija", tada je to funkcija čija je domena podskup skupa $\mathbb{N}^k$, a kodomena skup $\mathbb{N}$. Ako kažemo "funkcija", onda mislimo na neku k-mjesnu funkciju.

Totalna funkcija je funkcija čija je domena skup $\mathbb{N}^k$. Ako funkcija nije totalna, onda kažemo da je **parcijalna**.

Kada kažemo relacija, onda mislimo na neku k-mjesnu relaciju na prirodnim brojevima $R \subseteq \mathbb{N}^k$. Ako je R relacija, tada činjenicu da $\vec{x} \in R$ označavamo sa $R(\vec{x})$.

Karakterističnu funkciju relacije R označavamo sa χ_R, a definiramo je ovako:

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

Za relaciju R ćemo reći da je izračunljiva ako joj je pripadna karakterična funkcija izračunljiva. Također, primjetimo da je χ_R totalna za svaku relaciju R.

Nadalje, neka su U i V neki izrazi, te $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$ neka funkcija:

1. Činjenicu da je izraz U nedefiniran za neki $\vec{x} \in \mathbb{N}^k$ označavamo sa $U(\vec{x}) \uparrow$. U suprotnom, pišemo $U(\vec{x}) \downarrow$. Analogno, ako je f parcijalna funkcija, te $\vec{x} \in \mathbb{N}^k$ nije u domeni funkcije f, tada to označavamo sa $f(\vec{x}) \uparrow$. Ako pak $\vec{x}$ jest u domeni funkcije f, onda to označavamo sa $f(\vec{x}) \downarrow$.

2. Sa $U \simeq V$ (relacija "parcijalno jednako") označavamo činjenicu da za svaku uređenu k-torku prirodnih brojeva vrijedi jedno od sljedećeg:

a) $U(\vec{x})\downarrow, V(\vec{x})\downarrow$ i $U(\vec{x}) = V(\vec{x})$

b) $U(\vec{x})\uparrow$ i $V(\vec{x})\uparrow$

1.3 RAM-stroj

U ovom dijelu ćemo reći što je RAM-stroj, te ćemo ustvari dati njegovu opisnu definiciju, kako bi pojam RAM-stroja bio što jasniji. Definirat ćemo pojam RAM-izračunljivu funkciju. Sve ovo nužno je definirati kako bi kasnije mogli govoriti o generaliziranom RAM-stroju za parcijalno rekurzivne funkcionale.

RAM-stroj (eng. random access mashines) je apstraktan, idealiziran stroj s beskonačno velikom memorijom, koji nikada ne griješi. RAM-stroj predstavlja apstrakciju pojednostavljenog stvarnog sekvencijalnog računala.

RAM-stroj posjeduje prebrojivo mnogo registara, imenovanih prirodnim brojevima, od kojih svaki sadrži točno jedan po volji velik prirodan broj. Također, ima i vanjsku memoriju s izravnim pristupom koju može samo čitati. U literaturi postoje razne vrste definicija instrukcija RAM-stroja. U opisnoj definiciji vidjet ćemo da se za naša teorijska razmatranja uzimaju četiri tipa instrukcija kao "glavne". U nastavku dajemo opisnu definiciju RAM-stroja.

Definicija 1.3.1. *Osnovni dijelovi RAM-stroja su:*

- *registri*
- *spremnik za program*
- *brojač*

Za svaki prirodan broj k stroj ima registar koji ćemo označavati sa $\mathcal{R}_k$. Svaki $\mathcal{R}_k$ u svakom trenutku sadrži neki prirodan broj. Nadalje, u spremniku za program je smješten program. Svaki program je konačan niz instrukcija. Ako je n broj instrukcija u programu, tada ih redom numeriramo sa $1., 2., ..., n$. U brojaču se nalazi redni broj instrukcije koja se izvršava. Razmatramo četiri tipa instrukcija:

- *$INC\ \mathcal{R}_k$*

Kada se izvodi ova instrukcija, jednostavno se broj u registru $\mathcal{R}_k$ poveća za jedan, te se

broj u brojaču također poveća za jedan.

• *DEC* $\mathcal{R}_k$
Broj m je obavezno redni broj neke instrukcije u programu. Ako je broj u registru $\mathcal{R}_k$ *različit od nula, onda se broj u registru* $\mathcal{R}_k$ *smanji za jedan, te se broj u brojaču poveća za jedan. No, ako je broj u registru jednak nula, onda se samo broj u brojaču mijenja na m.*

• *GO TO m*
Broj m je obavezno redni broj neke instrukcije u programu. Kada se izvršava ova instrukcija, samo se mijenja broj u brojaču na m.

• *STOP*
Nakon ove instrukcije izračunavanje bezuvjetno staje.

Primjenu stroja vršimo na sljedeći način:

a) Stavljamo program u spremnik programa.

b) Upisujemo odgovarajuće brojeve u registre. Oni nam predstavljaju ulazne podatke. Ako se radi o programu s k ulaznih podataka, tada smatramo da su oni redom zapisani u registrima $\mathcal{R}_1, ..., \mathcal{R}_k$*. U brojaču je na početku broj 1 (to znači da se svaki program počinje izvršavati od prve instrukcije).*

c) Startamo stroj. Stroj tada počinje izvršavati instrukcije. U svakom koraku stroj izvršava instrukciju u programu čiji je redni broj u brojaču. Na kraju izvršenja instrukcije mijenja se broj u brojaču. Ako je izračunavanje došlo na instrukciju STOP, ili pak je broj u brojaču veći od svakog rednog broja instrukcija u programu, tada stroj staje. U tom slučaju je izlazni rezultat zapisan u registru $\mathcal{R}_0$*. Ako se to nikad ne dogodi stroj radi "vječno".*

Sada dajemo definiciju algoritma, te ćemo strogo definirati kada neki algoritam izračunava neku funkciju $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$. Zatim ćemo dati definiciju pojma RAM-izračunljive fu-
nkcije.

Definicija 1.3.2. *Za svaki program P za RAM-stroj, te za svaki* $k \in \mathbb{N}$ *uređeni par (P,k) nazivamo* ***algoritam****, te ga označavamo sa* A_k^P*.*

Definicija 1.3.3. *Neka je* $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$*, te* A_k^P *neki algoritam. Kažemo da* A_k^P ***izračunava*** *f ako za sve prirodne brojeve* $x_1, ..., x_k$ *vrijedi da je* $(x_1, ..., x_k) \in \mathbb{N}^k$ *ako i samo ako RAM-stroj s programom P u spremniku i s* $(x_1, ..., x_n)$ *kao ulaznim podacima stane, te je u tom*

slučaju na kraju rada stroja u registru $\mathcal{R}_0$ zapisan broj $f(x_1,...,x_k)$.
*Kažemo da je funkcija f **RAM-izračunljiva** ako postoji algoritam koji je izračunava. Nadalje, kažemo da je relacija R **RAM-izračunljiva** ako je njezina karakteristična funkcija RAM-izračnljiva.*

Često ćemo i reći da program P izračunava funkciju f, i to u situacijama kada nije potrebno naglašavati mjesnost ulaznih podataka. Ovdje navodimo neke rezultate iz kolegija Izračunljivost koji su nam važni za daljnja razmatranja.

Propozicija 1.3.4. *Svaka parcijalno rekurzivna funkcija je RAM-izračunljiva.*

Kako bismo kasnije mogli definirati generalizirani RAM-stroj (*GRAM-stroj*), bilo je nužno da smo ovdje naveli opisnu definiciju RAM-stroja, te definirali ustvari RAM-izračunljivu funkciju. Kako ćemo govoriti o *GRAM-stroj*, tako ćemo spomenuti i generalizirani Makro-stroj. Kako bi kasnije mogli reći što je generalizirani Makro-stroj, ekvivalentno sa RAM-strojem, i ovdje definiramo Makro-stroj na sljedeći način:

Definicija 1.3.5. *Za svaki program P za RAM-stroj uvodimo novu instrukciju koju označavamo sa P^* te ju nazivamo **makro** za P. **Makro-stroj** se sastoji od istih dijelova kao i RAM-stroj, te za svaki program P za RAM-stroj prepoznaje instrukciju P^*.*
Kada makro-stroj izvršava P^*, on zapravo izvršava program P s podacima koji se u tom trenutku nalaze u registrima. Ako izvršavanje programa P nikada ne stane, tada izvršavanje P^* nije dovršeno, te makro-stroj također ne staje. Ako izvršavanje P stane, tada se broj u brojaču u odnosu na broj u brojaču na početku izvršavanja P^* poveća za jedan, te makro-stroj jednostavno nastavlja izvršavati daljnje instrukcije programa.

Makro-izračunljiva funkcija definira se ekvivalentno kao i RAM-izračunljiva funkcija. Nadalje, valja uočiti kako je makro za P koji je gore definiran, zapravo pokrata za čitav niz osnovnih instrukcija za RAM-stroj, te da su ti novi tipovi instrukcija zapravo isto pokrati za niz osnovnih instrukcija za RAM-stroj.

Propozicija 1.3.6. *Klase RAM-izračunljivih funkcija i makro-izračunljivih funkcija su jednake.*

Primjer 1.3.7. *Neki makroi koje koristimo.*

- ***ZERO*** $\mathcal{R}_k$ *- Izjednačava broj u registru $\mathcal{R}_k$ sa nulom.*
- ***MOVE*** $\mathcal{R}_i$ ***TO*** $\mathcal{R}_j$ *- Ovaj makro broj iz registra $\mathcal{R}_i$ prepisuje u registar $\mathcal{R}_j$, pri čemu na kraju izvršavanja programa broj u registru $\mathcal{R}_i$ ostaje nepromijenjen. Naravno, i, j su međusobno različiti prirodni brojevi.*

- $f(\mathcal{R}_1,...,\mathcal{R}_k) \to \mathcal{R}_0$ - *Makro koji broj* $f(\mathcal{R}_1,...,\mathcal{R}_k)$ *smješta u registar* $\mathcal{R}_0$*, gdje je* f *neka funkcija.*

1.4 Rekurzivne funkcije

U ovom odijeljku ćemo do kraja definirati sve ostalo što nam je potrebno kako bismo mogli definirati parcijalno rekurzivne funkcionale. Definirat ćemo novu klasu izračunljivih funkcija. To su **parcijalno rekurzivne funkcije**.

Definicija 1.4.1. *Definiramo sljedeće funkcije:*

1. $Z : \mathbb{N} \to \mathbb{N}$ *sa* $Z(x) = 0$ *i nazivamo je* ***nul-funkcija****.*

2. $Sc : \mathbb{N} \to \mathbb{N}$ *sa* $Sc(x) = x + 1$ *i nazivamo je* ***funkcija sljedbenika****.*

3. $I_k^n : \mathbb{N}^n \to \mathbb{N}$ $(n \in \mathbb{N} \setminus \{0\},\ k \in \{1,\dots,n\})$*, definirana sa* $I_k^n(x_1,\dots,x_n) = x_k$*, i nazivamo je* ***projekcija****.*

Funkcije Z*,* Sc*,* $I_k^n : \mathbb{N}^n \to \mathbb{N}$ $(n \in \mathbb{N} \setminus \{0\},\ k \in \{1,\dots,n\})$ *nazivamo* ***inicijalne funkcije****.*

Definicija 1.4.2. *Neka je* G *neka* n*-mjesna funkcija. Neka su* $H_1,\dots,H_n$ k*-mjesne funkcije. Kažemo da je* k*-mjesna funkcija* F *definirana pomoću* ***kompozicije*** *funkcija ako za svaki* $\vec{x} \in \mathbb{N}^k$ *vrijedi:*

$$F(\vec{x}) \simeq G(H_1(\vec{x}),\dots,H_n(\vec{x})).$$

Definicija 1.4.3. *Neka je* G *totalna* k*-mjesna funkcija. Neka je* H *totalna* $(k+2)$*-mjesna funkcija. Kažemo da je* $(k+1)$*-mjesna funkcija* F *definirana pomoću* ***primitivne rekrzije*** *ako za svaki* $\vec{x} \in \mathbb{N}^k$*, te za svaki* $y \in \mathbb{N}$ *vrijedi:*

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

Definicija 1.4.4. *Neka je* f $(k+1)$*-mjesna funkcija. Kažemo da je funkcija* g *definirana pomoću* μ***-operatora*** *ako za svaki* $\vec{x} \in \mathbb{N}^k$ *vrijedi:*

$$g(\vec{x}) \simeq \begin{cases} min\{y \in \mathbb{N} | (\forall z < y) f(\vec{x},z) \downarrow \wedge f(\vec{x},y) \simeq 0\}, & \text{ako minimum postoji} \\ \uparrow, & \text{inače} \end{cases}$$

Funkciju g *obično označavamo sa* $\mu y(f(\vec{x},y) \simeq 0)$

Definicija 1.4.5. *Najmanja klasa funkcija koja sadrži sve inicijalne funkcije, te je zatvorena na kompoziciju i primitivnu rekurziju, zove se klasa* ***primitivno rekurzivnih funkcija****. Najmanja klasa funkcija koja sadrži sve inicijalne funkcije, te je zatvorena na kompoziciju, primitivnu rekurziju i μ-operator, zove se klasa* ***parcijalno rekurzivnih funkcija****.*
Funkcija iz klase parcijalno rekurzivnih funkcija koja je totalna zove se ***rekurzivna funkcija****. Nadalje, kažemo da je* ***relacija rekurzivna*** *ako je njezina karakteristična funkcija rekurzivna. Posebno, kažemo da je* ***skup rekzurzivan*** *ako je njegova karakteristična funkcija rekurzivna.*

Iz prethodne definicije možemo zaključiti da je svaka primitivno rekurzivna funkcija ujedno i rekurzivna, pa je onda i parcijalno rekurzivna. Prije iskaza Kleenijevog teorema o normalnoj navodimo jednu karakterizaciju parcijalno rekurzivnih funkcija koja će nam kasnije poslužiti za definiciju parcijalno rekurzivnih funkcionala.

Propozicija 1.4.6. *Funkcija $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$ je parcijalno rekurzivna ako i samo ako postoji konačan niz funkcija $g_1, \ldots, g_k$ $(k \geq 1)$ takav da je $f(\vec{x}) \simeq g_k(\vec{x})$, te za svaki $i \leq k$ vrijedi jedno od:*

1. *g_i je inicijalna funkcija.*

2. *Postoje $l, l_1, \ldots, l_n \leq i$ takvi da je g_i definirana pomoću kompozicije funkcija g_l i $g_{l_1}, \ldots, g_{l_n}$, tj. vrijedi $g_i(\vec{x}) \simeq g_l(g_{l_1}(\vec{x}), \ldots, g_{l_n}(\vec{x}))$*

3. *Postoje $l, m \leq i$ takvi da je g_i definirana pomoću primitivne rekurzije iz g_l i g_m, tj. vrijedi*

$$g_i(0, \vec{x}) = g_l(\vec{x})$$
$$g_i(y+1, \vec{x}) = g_m(g_i(y, \vec{x}), y, \vec{x}).$$

4. *Postoji $l < i$ takav da je g_i definirana pomoću μ-operatora iz funkcije g_l, tj. vrijedi*

$$g_i(\vec{x}) \simeq \begin{cases} min\{y \in \mathbb{N} | (\forall z < y) g_l(\vec{x}, z) \downarrow \wedge g_l(\vec{x}, y) \simeq 0\}, & \textit{ako minimum postoji} \\ \uparrow, & \textit{inače} \end{cases}$$

Dokaz. Označimo sa $\mathcal{P}$ klasu svih funkcija za koje postoji gore navedeni niz. Dovoljno je dokazati sljedeće:

a) Sve funkcije iz $\mathcal{P}$ su parcijalno rekurzivne.

b) $\mathcal{P}$ sadrži sve inicijalne funkcije.

c) $\mathcal{P}$ je zatvorena na kompoziciju, primitivnu rekurziju i μ-operator.

Tvrdnja *b*) je trivijalna.
Tvrdnju *a*) dokazujemo indukcijom po duljini niza $g_1, \ldots, g_k$. Za $k = 1$ imamo neku od inicijalnih funkcija. Svaka inicijalna funkcija je parcijalno rekurzivna, pa tvrdnja vrijedi. Pretpostavimo da su sve funkcije iz $\mathcal{P}$ za koje postoji niz duljine k parcijalno rekurzivne. Neka je g funkcija za koju postoji niz $g_1, \ldots, g_{k+1} \simeq g$.
Imamo dva slučaja:

(*i*) funkcija g_{k+1} je inicijalna funkcija, te je stoga i parcijalno rekurzivna.

(*ii*) Postoje funkcije $g_{l_1}, \ldots, g_{l_n}$, za $l_1, \ldots, l_n \leq k$ takve da je funkcija g_{k+1} definirana pomoću kompozicije, primitivne rekurzije ili μ-operatora pomoću funkcija $g_{l_1}, \ldots, g_{l_n}$. Po pretpostavci indukcije, funkcije $g_{l_1}, \ldots, g_{l_n}$ su parcijalno rekurzivne, te su stoga po definiciji zatvorene na kompoziciju, primitivnu rekurziju i μ-operator. Iz toga slijedi da je i funkcija g_{k+1} parcijalno rekurzivna.

Preostaje dokazati tvrdnju *c*). Neka su $g^1, \ldots, g^n \in \mathcal{P}$. Iz iskaza propozicije tada slijedi da za svaku od tih funkcija postoje nizovi $(g^1_1, \ldots, g^1_{k_1}), \ldots, (g^n_1, \ldots, g^n_{k_n})$. Neka je g funkcija definirana pomoću kompozicije, primitivne rekurzije ili μ-operatora pomoću funkcija $g^1, \ldots, g^n$. Očito je $g^1_1, \ldots, g^1_{k_1}, \ldots, g^n_1, \ldots, g^n_{k_n}, g$ upravo traženi niz za g. Slijedi $g \in \mathcal{P}$, čime smo dokazali tvrdnju.

□

Poglavlje 2

Parcijalno rekurzivni funkcionali

U prethodnom poglavlju smo govorili o RAM-strojevima, rekurzivnosti, te definirali parcijalno rekurzivne funkcije. U ovom poglavlju prvo ćemo dati motivaciju za jedan novi pojam, a to je pojam **funkcionala**. Napravit ćemo generalizaciju parcijalno rekurzivnih funkcija na parcijalno rekurzivne funkcionale. Opisnu definiciju RAM-stroja ćemo proširiti na definiciju GRAM-stroja, te dati definiciju **GRAM-izračunljive funkcije**. Nadalje ćemo teorem o normalnoj formi za parcijalno rekurzivne funkcije proširiti teoremom o normalnoj formi za parcijalno rekurzivne funkcionale, te dati dokaze. Također ćemo uvesti pojam ograničenog parcijalno rekurzivnog funkcionala. Iskazat ćemo i dokazati prvi teorem rekurzije. Definirat ćemo pojam fiksne točke, te iskazati i dokazati teorem o fiksnoj točki.

2.1 Motivacija

Vidjeli smo da svakoj parcijalno rekurzivnoj funkciji možemo pridružiti broj $e \in \mathbb{N}$, koji se naziva indeks funkcije. Kako za svaku parcijalno rekurzivnu funkciju postoji beskonačno mnogo programa koji je izračunavaju, tako postoji i beskonačno mnogo indeksa. To znači da taj indeks nije jedinstven za određenu funkciju. Također, za svaki $e \in \mathbb{N}$ postoji parcijalno rekurzivna funkcija čiji je e indeks. Ako uzmemo izraz $\{e\}^k(\vec{x}) \simeq U(\mu y T_n(e, \vec{x}, y))$ (vidi [2]), te stavimo da je $F_e(\vec{x}) \simeq \{e\}$, onda vidimo da je to upravo funkcija $\{e\}^k(\vec{x}) \simeq U(\mu y T_n(F_e(\vec{x}), \vec{x}, y))$. Očito je i funkcija $(e, \vec{x}) \mapsto \{e\}^k(\vec{x})$ također parcijalno rekurzivna. Iz prethodnoga slijedi da su parcijalno rekurzivne i slijedeće funkcije:

a) $Int^k(e, \vec{x}) \simeq \{e\}^k(\vec{x})$. Ova funkcija računa funkciju s indeksom e u zadanoj točki $\vec{x}$. Budući je funkcija parcijalno rekurzivna ako i samo ako je RAM-izračunljiva, slijedi da možemo napraviti RAM-stroj koji ustvari simulira rad samog RAM-stroja.

b) $\beta(k, l, x) \simeq \{i\}(\{j\}(x))$. Ako funkciji β kao prva dva argumenta proslijedimo indekse nekih

parcijalno rekurzivnih funkcija f i g, tada će β na x djelovati kao kompozicija funkcija f i g.

c) Pomoću primitivne rekurzije definiramo funkciju ϕ ovako:

$$\begin{aligned} \phi(e,0) &\simeq 1 \\ \phi(y+1,e) &\simeq \phi(y,e) * \langle\{e\}(y)\rangle, \end{aligned}$$

gdje je s $*$ označena konkatenacija nizova. Kada bi e bio indeks neke rekurzivne (totalne) funkcije F te $y \in \mathbb{N}$, onda bi funkcija $\phi(e,y)$ zapravo računala prvih y vrijednosti funkcije F.

Cilj gornjih primjera bio je pokazati kako postoje parcijalno rekurzivne funkcije koje preko indeksa kao argumente mogu primati neke druge parcijalno rekurzivne funkcije.
Prethodno spomenuto daje nam motivaciju za uvođenje pojma funkcionala, te definiciju parcijalno rekurzivnih funkcionala.

2.2 Pojam funkcionala i definicija parcijalno rekurzivnih funkcionala

U ovom odjeljku generaliziramo pojam parcijalno rekurzivne funkcije definirajući pojam funkcionala i parcijalno rekurzivnih funkcionala. Reći ćemo kako odrediti da je neki funkcional parcijalno rekurzivan, te sve ilustrirati kroz nekoliko primjera.

Definicija 2.2.1. ***Funkcional*** *je funkcija čiji su argumenti prirodni brojevi ili funkcije, a vrijednost poprima na skupu prirodnih brojeva.*
Funkcional je ***totalan*** *ako je definiran kad god su mu funkcijski argumenti totalni.*
Ako promatramo funkcionale samo na totalnim funkcijama onda govorimo o ***ograničenim*** *funkcionalima.*

Primjer 2.2.2. *Neka je* $A = \{f | f : 2\mathbb{N} \to \mathbb{N}\}$. *Neka je* $G = \{g | g : 2\mathbb{N}+1 \to \mathbb{N}\}$. *Definiramo funkcional* $F : A \times B \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *ovako:*

$$F(f,g,x,y) = 3f(x) + 5g(f(x+y)).$$

Primjer 2.2.3. *Neka je* $S \subseteq \mathbb{N}^2$ *zadan. Neka je* $A = \{f \mid f : S \to \mathbb{N}\}$. *Definiramo funkcional* $G : A \times \mathbb{N}^3 \to \mathbb{N}$ *ovako:*

$$G(f,x_1,x_2,x_3) = 2f(x_2,x_3) + x_1.$$

Kada smo u prethodnom poglavlju govorili o funkcijama, na početku smo rekli što je to proces izračunavanja te smo intuitivno rekli što znači kada je neka funkcija izračunljiva.

Za funkcionale također vrijedi taj analogon. Imamo početne objekte koje nazivamo ulaznim podacima i koji nam ovdje predstavljaju konačan niz ulaznih funkcija $g_1, \ldots, g_m$ te konačan broj ulaznih parametara $\vec{x} \in \mathbb{N}^k$. Primjenom određenog skupa pravila (algoritam) na navedene ulazne podatke dobivamo kranji rezultat kojeg nazivamo izlazni podatak. Taj proces zovemo izračunavanje. Funkcional F je **izračunljiv** ako izračunavanje s ulaznim funkcijama $g_1, \ldots, g_m$ te ulaznim parametrima $\vec{x} \in \mathbb{N}^k$ stane u konačno mnogo koraka ako i samo ako su funkcije $g_1, \ldots, g_m$ i brojevi $\vec{x}$ u domeni funkcionala F. Izlazni rezultat je u tom slučaju jednak $F(g_1, \ldots, g_m, \vec{x})$.

Sada ćemo definirati operacije kompozicije, primitivne rekurzije i μ-operatora za parcijalno rekurzivne funkcionale te ćemo dati definiciju parcijalno rekurzivnih funkcionala.

Definicija 2.2.4. *Neka su $G, H_1, \ldots, H_n$ funkcionali. Kažemo da je funkcional F definiran pomoću kompozicije funkcionala $G, H_1, \ldots, H_n$ ako za sve funkcije $g_1, \ldots, g_m$ te za sve parametre $\vec{x} \in \mathbb{N}^k$ vrijedi:*

$$F(g_1, \ldots, g_m, \vec{x}) \simeq G(H_1(g_1, \ldots, g_m, \vec{x}), \ldots, H_n(g_1, \ldots, g_m, \vec{x})).$$

Definicija 2.2.5. *Neka su G i H funkcionali. Kažemo da je funkcional F definiran pomoću primitivne rekurzije iz funkcionala G i H ako za sve funkcije $g_1, \ldots, g_m$, sve parametre $\vec{x} \in \mathbb{N}^k$ te za sve $y \in \mathbb{N}$ vrijedi:*

$$\begin{aligned} F(g_1, \ldots, g_m, \vec{x}, 0) &= G(g_1, \ldots, g_m, \vec{x}) \\ F(g_1, \ldots, g_m, \vec{x}, y+1) &= H(F(g_1, \ldots, g_m, y, \vec{x}), y, \vec{x}) \end{aligned}$$

Definicija 2.2.6. *Neka je F funkcional. Tada kažemo da je funkcional G definiran pomoću μ-operatora ako za sve funkcije $g_1, \ldots, g_m$, te za sve parametre $\vec{x} \in \mathbb{N}^k$ vrijedi:*

$$G(g_1, \ldots, g_m, \vec{x}) \simeq \begin{cases} min\{y \in \mathbb{N} | (\forall z < y) F(g_1, \ldots, g_m, \vec{x}, z) \downarrow \wedge \\ F(g_1, \ldots, g_m, \vec{x}, y) \simeq 0\}, & \textit{ako min postoji} \\ \uparrow, & \textit{inače} \end{cases}$$

Definicija 2.2.7. *Najmanja klasa funkcionala koja sadrži sve inicijalne funkcije te je zatvorena na kompoziciju, primitivnu rekurziju i μ-operator naziva se klasa **parcijalno rekurzivnih funkcionala**. Funkcional iz klase parcijalno rekurzivnih funkcionala zove se **parcijalno rekurzivan funkcional**. Za parcijalno rekurzivan funkcional F kažemo da je **rekurzivan** ako je definiran za sve (totalne) funkcije $g_1, \ldots, g_m$, te za sve parametre $\vec{x} \in \mathbb{N}^k$. Kažemo da je funkcional F (parcijalno) rekurzivno **ograničen** funkcional ako se u njegovoj domeni nalaze samo totalne funkcije g_i, $(i \leq m)$.*

Napomena 2.2.8. *Primijetimo kako smo u prethodnim definicijama zahtijevali da svi funkcionali u nizu budu ograničeni s točno m funkcijskih varijabli. To smo napravili samo radi jednostavnijeg zapisa. Naime, neka je $F(g_1, \ldots, g_n, \vec{x})$ parcijalno rekurzivan funkcional. Tada je očito parcijalno rekurzivan i funkcional $G(g_1, \ldots, g_{n+k}, \vec{x}) \simeq F(g_1, \ldots, g_n, \vec{x})$. Iz ovoga vidimo da broj funkcijskih varijabli ne mora biti isti za sve funkcionale u nizu.*

Definicija 2.2.9. ***Relacija na funkcijama i brojevima** je podskup skupa $S^m \times \mathbb{N}^n$, gdje je S neki skup funkcija te $m, n \in \mathbb{N}$. Kažemo da je relacija R na funkcijama i brojevima parcijalno rekurzivna ako je njezin karakteristični funkcional χ_R parcijalno rekurzivan.*
*Kao i kod funkcionala, možemo govoriti o rekurzivno **ograničenim** relacijama na funkcijama i brojevima.*

Napomena 2.2.10. *Kako bi dokazali da je neki funkcional parcijalno rekurzivan, dovoljno je pokazati da postoji niz funkcionala $F_1, \ldots, F_k$ takav da svaki funkcional F_i $(i \leq k)$ iz niza zadovoljava jednu od tvrdnji iz definicije za kompoziciju, primitivnu rekurziju ili μ-operator ili je pak sam parcijalno rekurzivan. Traženi niz funkcionala tada dobivamo konkatenacijom pripadnih nizova svih funkcionala F_i za koje smo pokazali da su parcijalno rekurzivni. Na kraju dodamo ostale funkcionale između $F_1, \ldots, F_k$.*

Primjer 2.2.11.

a) *Funkcional aplikacije $Ap(g, \vec{x}) \simeq g(\vec{x})$ je parcijalno rekurzivan.*

b) *Funkcional kompozicije $Comp(f, g, x) \simeq f(g(x))$ je parcijalno rekurzivan. Niz iz definicije za $Comp(f, g, x)$ je:*

1. $F_1(f, g, x) \simeq f(x)$ - *aplikacija*

2. $F_2(f, g, x) \simeq g(x)$ - *aplikacija*

3. $F_3(f,g,x) \simeq F_2(f,g,F_1(f,g,x))$ - *kompozicija* F_1 *i* F_2

c) *Funkcional* $Vals(f,y)$ *koji je definiran s*

$$Vals(f,0) \simeq 1$$
$$Vals(f,y+1) \simeq Vals(f,y) * \langle f(y) \rangle$$

je parcijalno rekurzivan. Odgovarajući niz funkcionala je:

1. $F_1(f,y) \simeq f(y)$

2. $F_2(x,y) \simeq x * \langle y \rangle$

3. $F_3(x,y) \simeq x$

4. $F_4(x,y) \simeq y$

5. $F_5(f,x,y) \simeq F_1(f,F_4(x,y))$

6. $F_6(f,x,y) \simeq F_2(F_3(x,y),F_5(f,x,y))$

7. $F_7(f) \simeq 1$

8. $F_8(f,y) = \pi(F_7,F_6)(f,y)$,

gdje je $\pi(G,H)$ *operator primitivne rekurzije na funkcionalima G i H definiran na sljedeći način:*

$$\pi(G,H)(f_1,\ldots,f_k,\vec{x},0) \simeq G(f_1,\ldots,f_k,\vec{x})$$
$$\pi(G,H)(f_1,\ldots,f_k,\vec{x},y+1) \simeq H(f_1,\ldots,f_k,\pi(G,H)(f_1,\ldots,f_k,\vec{x},y),\vec{x},y).$$

Napomena 2.2.12. *Uočimo kako parcijalno rekurzivni funkcionali nisu općenito izračunljivi. Primjerice, kao funkcijski argument možemo proslijediti neku funkciju koja nije parcijalno rekurzivna. Na primjer, funkcional aplikacije Ap može odgovoriti na pitanje da li je par-*

cijalno rekurzivna funkcija s indeksom e definirana za neku vrijednost $\vec{x}$ ako mu kao prvi argument proslijedimo funkciju χ_H gdje je H funkcija $H = \{(e,x) \mid \{e\}(\vec{x})\downarrow\}$.

2.3 Izračunavanje s prorokom

U kontekstu izračunljivosti, **prorok** predstavlja jedan apstraktni entitet kojeg koristimo za rješavanje "problema odlučivanja" ili funkcijskih problema. Problem odlučivanja može biti iz bilo koje klase složenosti. Možemo koristiti čak i neriješive probleme kao što je to Halting problem ([2]). Možemo ga zamisliti kao povezivanje RAM-stroja kojeg smo definirali u prethodnom poglavlju sa "crnom kutijom" (vidi [3]). Nju nazivamo prorok, te s njim u jednoj operaciji rješavamo određene probleme odlučivanja. Prorok nam daje rješenje za svaku instancu danog problema izračunavanja.

Primjer 2.3.1.

a) Neka je problem odlučivanja dan skupom A prirodnih brojeva $\{1,2,\ldots,n\}$, Instanca tog problema je proizvoljan prirodan broj k. Prorok u ovom slušaju može dati odgovor DA ako je $k \in A$, odnosno NE ako $k \notin A$.

b) Neka je $f : \mathbb{N} \to \mathbb{N}$ proizvoljna funkcija te neka je x proizvoljan ulaz. Prilikom izračunavanja s RAM-strojem, možemo poslati proroku pitanje je li funkcija f definirana u točki x. On će opet odgovoriti sa DA u slučaju da je f definirana u x, odnosno NE ako to nije slučaj.

U našim razmatranjima, izračunavanje s prorokom značit će proširenje RAM-stroja instrukcijom $ASK\mathcal{O}_j$. Ako izračunavanje s RAM-strojem dođe dođe do nekog podproblema kojeg ne zna riješiti, onda RAM-stroj ne može nastaviti s izračunavanjem (njegov problem nije izračunljiv). U tom slučaju on će za pomoć pitati jednog od proroka $\mathcal{O}_j$, koji mu sigurno daje rješenje njegovog problema, te RAM-stroj može nastaviti s radom.
Kada smo dali ideju izračunavanja s prorokom, možemo proširiti definiciju RAM-stroja iz prethodnog poglavlja na GRAM-stroj za parcijalno rekurzivne funkcionale.

2.4 GRAM-stroj i GRAM-izračunljive funkcije

Ovdje je cilj dati definiciju modificiranog RAM-stroja za izračunavanje parcijalno rekurzivnih funkcionala. Definiramo GRAM-izračunljiv funkcional i dajemo primjere.

Definicija 2.4.1. *Generalizirani RAM-stroj (GRAM-stroj) je idealizirano računalo koje se sastoji od sljedećih dijelova:*

- *prebrojivo mnogo registara koji su redom označeni sa* $\mathcal{R}_1, \mathcal{R}_2, \ldots$. *Za svaki prirodan broj k stroj ima registar kojeg označavamo s* $\mathcal{R}_k$. *Svaki registar* $\mathcal{R}_k$ *u svakom trenutku rada stroja sadrži neki prirodan broj.*

- *spremnik za program u kojem je smješten program GRAM-stroja. Svaki program predstavlja konačan niz instrukcija. Ako je n broj instrukcija u programu, tada ih redom numeriramo* $1, 2, \ldots, n$.

- *brojač u kojemu se nalazi redni broj instrukcije koju program izvršava.*

- *prebrojivo mnogo spremnika za posebne funkcije* $\mathcal{O}_1, \ldots$. *Svaki od spremnika za posebne funkcije* $\mathcal{O}_j$ *sadrži neku funkciju* $f : S \subseteq \mathbb{N}^k \to \mathbb{N}, k \in \mathbb{N}$, *pri čemu* S *i* k *ne moraju biti jednaki za sve spremnike.*

Promatramo pet tipova instrukcija:

- $INC\ \mathcal{R}_k$

- $DEC\ \mathcal{R}_k, m$

- $GO\ TO$

- $STOP$

- $ASK\mathcal{O}_j$

Gornje instrukcije smo već definirali prilikom definicije RAM-stroja u prethodnom poglavlju. Instrukcija $ASK\ \mathcal{O}_j$ *u registar* $\mathcal{R}_0$ *stavlja broj koji je dobiven primjenom funkcije koja se nalazi u spremniku* $\mathcal{O}_j$ *na vrijednosti koje se nalaze u registrima u tom trenutku te povećava broj u brojaču za jedan.*

Rad stroja opisujemo na sljedeći način:

a) U spremnik za program stavljamo GRAM-program i u brojač stavljamo vrijednost 1 (svaki program se počinje izvršavati od prve instrukcije).

b) U registre $\mathcal{R}_1, \ldots, \mathcal{R}_n$ *upisujemo ulazne podatke (prirodni brojevi). U spremnik za posebne funkcije* $\mathcal{O}_1, \mathcal{O}_2 \ldots, \mathcal{O}_m$ *stavljamo ulazne funkcije. U svim ostalim registrima sma-*

tramo da je zapisana vrijednost 0*, a u svim ostalim spremnicima za posebne funkcije prazna funkcija* ($\emptyset$).

Tada možemo startati stroj.

Stroj počinje izvršavati instrukcije čiji se redni broj nalazi u brojaču. Izvršavanje pojedinih instrukcija smo definirali u prethodnom poglavlju kada smo definirali RAM-stroj. Ovdje navodimo način izvršavanja instrukcije ASK ovako:

- *Ako je pročitana instrukcija $ASK\ \mathcal{O}_j$ i ako je u spremniku $\mathcal{O}_j$ spremljena funkcija f s k argumenata tada se u registar $\mathcal{R}_0$ sprema vrijednost koja je dobivena primjenom funkcije f na vrijednosti zapisane u $\mathcal{R}_1, \ldots, \mathcal{R}_k$. Ako f nije definirana u traženoj točki, onda definiramo da GRAM-stroj nikad ne staje.*

GRAM-stroj će stati ako vrijednost u brojaču nije redni broj neke instrukcije programa ili ako je izvršena instrukcija $STOP$. U tom slučaju je izlazni rezultat zapisan u registru $\mathcal{R}_0$.

Slijede jednostavni primjeri GRAM-programa.

Primjer 2.4.2.

a) Pretpostavimo da je u spremniku $\mathcal{O}_1$ spremljena funkcija koja je definirana sa $f(x,y) = (x+1)+(y+1)$, $(x,y \in \mathbb{N})$. Sljedeći program za GRAM-stroj prvo povećava sadržaje registara $\mathcal{R}_1$ i $\mathcal{R}_2$ za jedan te primjenom definirane funkcije f na sadržaje registara $\mathcal{R}_1$ i $\mathcal{R}_2$ u registar $\mathcal{R}_0$ zapisuje broj $x+y+4$.

1. $INC\ \mathcal{R}_1$
2. $INC\ \mathcal{R}_2$
3. $ASK\ \mathcal{O}_1$
4. $STOP$

b) Neka je $f : \mathbb{N} \to \mathbb{N}$ funkcija definirana ovako:

$$f(x) = \begin{cases} x, & \text{ako vrijedi da je } x \text{ paran} \\ \text{nedefinirano}, & \text{inače} \end{cases}$$

Sljedeći program za GRAM-stroj prvo postavlja vrijednost registra $\mathcal{R}_1$ na nulu te ga zatim povećava za jedan te se u registru $\mathcal{R}_1$ nalazi broj jedan koji je neparan. Funkcija koja se nalazi u spremniku $\mathcal{O}_1$ nije definirana za dani ulazni podatak i GRAM-stroj neće stati.

1. $DEC\ \mathcal{R}_1, 3$

2. $GO\ TO\ 1$

3. $INC\ R_1$

4. $ASK\ \mathcal{O}_1$

c) Neka je $f : \mathbb{N}^2 \to \mathbb{N}$ funkcija definirana sa

$$f(x,y) = \begin{cases} 1, & \textit{ako vrijedi } x > y \\ 0, & \textit{inače} \end{cases}$$

i neka se nalazi u spremniku $\mathcal{O}_1$. Sljedeći GRAM program povećava sadržaj registra $\mathcal{R}_1$ za dva te postavlja sadržaj registra $\mathcal{R}_2$ na 0. Zatim primjenom definirane funkcije na sadržaje registara $\mathcal{R}_1$ i $\mathcal{R}_2$ upisuje u registar $\mathcal{R}_0$ broj 1.

1. $INC\ \mathcal{R}_1$

2. $DEC\ \mathcal{R}_2, 4$

3. $GO\ TO\ 2$

4 $INC\ \mathcal{R}_1$

5. $ASK\ \mathcal{O}_1$

6. $STOP$

Sada ćemo definirati GRAM-izračunljiv funkcional te dati nekoliko primjera GRAM-izračunljivih funkcionala.

Definicija 2.4.3. *Neka je P program za GRAM-stroj i $F(g_1, \ldots, g_m, \vec{x})$ neki funkcional. Kažemo da program P za GRAM-stroj izračunava funkcional F ako vrijedi:*

1. *Program P s ulaznim fukcijama $g_1, \ldots, g_m$ i ulaznim parametrima $\vec{x}$ staje*

ako i samo ako $F(g_i, \ldots, g_m, \vec{x}) \downarrow$.

2. *Ako P stane onda je izlazni rezultat* $F(g_1, \ldots, g_m, \vec{x})$.

Kažemo da je funkcional F ***GRAM-izračunljiv*** *ako postoji program koji ga izračunava.*

Primjer 2.4.4.

a) *Funkcional aplikacije* $Ap(f, \vec{x}) \simeq f(\vec{x})$ *je GRAM-izračunljiv. Program koji ga izračunava je*

1. $AS\,K\ \mathcal{O}_1$

b) *Funkcional kompozicije definiran je sa:*

$$Comp(f, g, x) \simeq f(g(x)).$$

Program koji ga izračunava je:

1. $AS\,K\ \mathcal{O}_2$
2. $DEC\ \mathcal{R}_1, 4$
3. $GO\ TO\ 2$
4. $DEC\ \mathcal{R}_0, 7$
5. $INC\ \mathcal{R}_1$
6. $GO\ TO\ 4$
7. $AS\,K\ \mathcal{O}_1$

c) *GRAM-stroj sadrži sve dijelove kao i RAM-stroj pa je svaki RAM-program ujedno i GRAM-program, pa je svaka RAM-izračunljiva funkcija i GRAM-izračunljiva. Iz propozicije 1.3.4 slijedi da je svaka parcijalno rekurzivna funkcija GRAM-izračunljiva.*

GRAM-stroj može izračunati i funkcije koje nisu izračunljive. Instrukcija $AS\,K\ \mathcal{O}_j$ omogućuje GRAM-stroju da izračuna svaku funkciju na prirodnim brojevima. To naravno znači da se GRAM-stroj ne može realizirati kao posebno računalo. RAM-stroj, s druge

strane, računa sve dok ne dođe do nekog podproblema kojeg ne zna riješiti te u takvoj situaciji ne može nastaviti s izračunom. Kada GRAM-stroj dođe u takvu situaciju, on za pomoć pita jednog od proroka $\mathcal{O}_j$ koji mu da rješenje njegovog problema te GRAM-stroj može normalno nastaviti s radom.

Prema analogonu iz prethodnog poglavlja i ovdje definiramo generalizirani makro-stroj.

Definicija 2.4.5. *Za svaki program P za GRAM-stroj uvodimo novu instrukciju i označavamo je s P^* i zovemo je makro za P. Generalizirani makro-stroj sastoji se iz istih djelova kao i GRAM-stroj te za svaki program P za GRAM-stroj prepoznaje instrukciju P^*. Kada generalizirani makro-stroj izvršava instrukciju oblika P^*, on zapravo izvršava program P s podacima koji se trnutno nalaze u registrima. Ako izvršavanje programa P nikada ne stane, tada kažemo da izvršavanje instrukcije P^* nije dovršeno te generalizirani makro-stroj također neće stati. Ako izvršavanje programa P stane, tada se broj u brojaču u odnosu na broj u brojaču na početku izvršavanja instrukcije P^* poveća za jedan te generalizirani makro-stroj normalno nastavlja izvršavanje daljnjih instrukcija u programu.*

Ova nam definicija omogućava korištenje makroa za GRAM-stroj.
GRAM-stroj kojeg smo definirali može izračunati svaki parcijalno rekurzivan funkcional. To dokazujemo u sljedećoj propoziciji.

Propozicija 2.4.6. *Svaki parcijalno rekurzivan funkcional je GRAM-izračunljiv.*

Dokaz. Neka je $F(g_1, \ldots, g_m, \vec{x})$ parcijalno rekurzivan funkcional. Neka je $F_1, \ldots, F_k$ pripadni nizovi funkcionala iz napomene 2.2.10. Tvrdnju ćemo dokazati indukcijom po duljini niza $F_1, \ldots, F_k$.

a) Za $k = 1$ imamo da je F_1 ili aplikacija ili jedna od inicijalnih funkcija. Njihova GRAM-izračunljivost tada slijedi iz primjera 2.2.11

b) Pretpostavimo sada da je svaki parcijalno rekurzivan funkcional s nizom duljine najviše k GRAM-izračunljiv. Neka je F funkcional s m funkcijskih i n brojevnih varijabli s nizom duljine $k+1$. Svaki od funkcionala F_i, $i \leq k$ je parcijalno rekurzivan, jer mu je pripadni niz $F_1, \ldots, F_i$. Tada je po pretpostavci indukcije i GRAM-izračunljiv pa postoji program P_i za GRAM-stroj koji ga izračunava. Funkcional $F \simeq F_{k+1}$ je funkcional aplikacije, inicijalna funkcija ili je nastao kompozicijom, primitivnom rekurzijom ili primjenom μ-operatora na iz prethodnih funkcionala niza. Ako je F funkcional aplikacije ili jedna od inicijalnih funkcija tada je i GRAM-izračunljiv. Ako je pak rezultat kompozicije, primitivne rekurzije ili μ-operatora onda dokazujemo da je F GRAM-izračunljiv tako što za svaku operaciju pišemo program za generalizirani makro-stroj koji ga izračunava. Prvo definirajmo makroe koje ćemo koristiti:

- $ZERO\ \mathcal{R}_i$-$\mathcal{R}_j$ – postavlja vrijednosti registara $\mathcal{R}_i, \ldots, \mathcal{R}_j$ na nulu.

- $COPY\ \mathcal{R}_i - \mathcal{R}_j\ TO\ \mathcal{R}_k$ – kopira vrijednosti registara $\mathcal{R}_i - \mathcal{R}_j$ u registre $\mathcal{R}_k, \ldots, \mathcal{R}_{k+j-i}$.

U svakom od programa P_i, $i \leq k$ se nalazi konačan broj instrukcija, pa time i konačan broj registara koje svaki program koristi. Označimo s L za jedan veći prirodan broj od najvećeg indeksa registra koji se koriste u instrukcijama programa P_i. Sada možemo dati tražene programe.

- Ako je funkcional F dobiven kompozicijom iz funkconala F_i i $F_{i_1}, \ldots, F_{i_j}$, tada jedan program za GRAM-stroj koji izračunava funkcional F izgleda ovako:

  1. $COPY\ \mathcal{R}_0 - \mathcal{R}_{L-1}\ TO\ \mathcal{R}_L$

  2. $P^*_{i_1}$

  3. $COPY\ \mathcal{R}_0 - \mathcal{R}_0\ TO\ \mathcal{R}_{2L}$

  4. $COPY\ \mathcal{R}_L - \mathcal{R}_{2L-1}\ TO\ \mathcal{R}_0$

  5. $P^*_{i_2}$

  6. $COPY\ \mathcal{R}_0 - \mathcal{R}_0\ TO\ \mathcal{R}_{2L+1}$

  7. $COPY\ \mathcal{R}_L - \mathcal{R}_{2L-1}\ TO\ \mathcal{R}_0$

  $\vdots$

  $(3j-1)$. P_{i_j}

  $3j$. $COPY\ \mathcal{R}_0 - \mathcal{R}_0\ TO\ \mathcal{R}_{2L+j-1}$

  $(3j+1)$. $ZERO\ \mathcal{R}_0 - \mathcal{R}_{L-1}$

  $(3j+2)$. $COPY\ \mathcal{R}_{2L} - \mathcal{R}_0\ TO\ \mathcal{R}_{2L+j-1}$

  $(3j+3)$. P^*_i

- Ako je funkcional F dobiven primitivnom rekurzijom iz F_i i F_j, tada jedan pro-

gram za GRAM-stroj koji izračunava funkcional F izgleda ovako:

1. $COPY\ \mathcal{R}_0 - \mathcal{R}_{L-1}\ TO\ \mathcal{R}_L$
2. $ZERO\ \mathcal{R}_0 - \mathcal{R}_{L-1}$
3. $COPY\ \mathcal{R}_{L+2} - \mathcal{R}_{2L-1}\ TO\ \mathcal{R}_1$
4. P_i^*
5. $DEC\ \mathcal{R}_{L+1}, 14$
6. $\mathcal{R}_1 - \mathcal{R}_{L-1}$
7. $COPY\ \mathcal{R}_0 - \mathcal{R}_0\ TO\ \mathcal{R}_1$
8. $ZERO\ \mathcal{R}_0 - \mathcal{R}_0$
9. $COPY\ \mathcal{R}_{L+2} - \mathcal{R}_{2L-2}\ TO\ \mathcal{R}_3$
10. $INC\ \mathcal{R}_L$
11. $COPY\ \mathcal{R}_L - \mathcal{R}_L\ TO\ \mathcal{R}_2$
12. P_j^*
13. $GO\ TO\ 5$
14. $STOP$

- Ako je funkcional F dobiven primjenom μ-operatora na funkcional F_i i ako F ima n brojevnih varijabli, tada jedan program za GRAM-stroj koji izračunava funkcional F izgleda ovako:

  1. $COPY\ \mathcal{R}_0 - \mathcal{R}_{L-1}\ TO\ \mathcal{R}_{\mathcal{L}}$
  2. P_i^*
  3. $DEC\ \mathcal{R}_0, 7$

4. $INC\ \mathcal{R}_{L+n+1}$

5. $COPY\ \mathcal{R}_L - \mathcal{R}_{2L-1}\ TO\ \mathcal{R}_0$

6. $GO\ TO\ 2$

7. $COPY\ \mathcal{R}_{L+n+1} - \mathcal{R}_{L+n+1}\ TO\ \mathcal{R}_0$

□

2.5 Teorem o normalnoj formi za parcijalno rekurzivne funkcionale

U ovom odijeljku ćemo iskazati te dati i skicu dokaza za Kleenijev teorem o normalnoj formi za parcijalno rekurzivne funkcije. Ta skica će nam poslužiti kao analogon za dokaz teorema o normalnoj formi za parcijalno rekurzivne funkcionale. Na kolegiju Izračunljivosti dan je detaljan dokaz ovog teorema i može se naći u [2]. Direktan dokaz ovog teorema također se može naći i u [1]. U prethodnom smo poglavlju definirali RAM-stroj i RAM-izračunljivu funkciju.
Dok budemo davali skicu dokaza, paralelno ćemo pokazati kako "efektivno kodirati" konačne nizove prirodnih brojeva te instrukcije RAM-stroja. To ćemo napraviti definirajući funkciju kodiranja konačnog niza te kod konačnog niza. Ilustrirat ćemo kodiranje kroz nekoliko primjera.

Teorem 2.5.1 (Kleenijev teorem o normalnoj formi)**.** *Postoji primitivno rekurzivna funkcija U, te za svaki $k \in \mathbb{N} \setminus \{0\}$ postoji primitivno rekurzivna funkcija T_k tako da za svaku parcijalno rekurzivnu funkciju f postoji broj e takav da vrijedi:*

1. $f(\vec{x}) \Leftrightarrow \exists y T_n(e, \vec{x}, y)$
2. $f(\vec{x}) \simeq U(\mu y T_n(e, \vec{x}, y))$

*pri čemu je $\mu y T_n(e, \vec{x}, y) = \mu y(1 \dot{-} \chi_R(\vec{x}) T_n(e, \vec{x}, y) \simeq 0)$. Broj e nazivamo **indeks** funkcije f.*

Sada dajemo definicije koda, te funkcije kodiranja.

Definicija 2.5.2. *Neka je* $\langle\cdot\rangle : \mathbb{N}^k \to \mathbb{N}$ *funkcija definirana ovako*

$$\langle x_1, \ldots, x_k \rangle = p_1^{x_1+1} \cdot \ldots \cdot p_k^{x_k+1}$$

gdje je p_i *i-ti po redu prost broj. Funkciju* $\langle\cdot\rangle$ *zovemo* ***funkcijom kodiranja*** *konačnog niza, a vrijednost* $\langle\vec{x}\rangle$ ***kod*** *konačnog niza* $\vec{x} \in \mathbb{N}^k$. *Posebno,* $\langle\rangle = 1$ *(kod praznog niza je jednak 1).*

Nadalje, instrukcije RAM-stroja kodiramo na sljedeći način:

a) $INC\ \mathcal{R}_k \mapsto \langle 0, k\rangle$

b) $DEC\ \mathcal{R}_k, m \mapsto \langle 1, k, m\rangle$

c) $GO\ TO\ m \mapsto \langle 2, m\rangle$

d) $STOP\ \mathcal{R}_k \mapsto \langle 3\rangle$

Sada kodiramo program P za RAM-stroj.
Ako su $y_1, \ldots, y_n$ redom kodovi instrukcija programa P, tada je kod programa P broj $e = \langle y_1, \ldots, y_N\rangle$. Na isti način možemo kodirati i rad RAM-stroja s programom P u spremniku te sa $\vec{x}$ kao ulaznim podacima. Za program P s kodom e i sa k ulaznih parametara vidimo da je dovoljno kodirati prvih $e + k$ registara RAM-stroja (ostali se očito i ne koriste). Za i-ti korak programa označimo sa r_j^i vrijednost registra $\mathcal{R}_j$. Tada je kod registra u i-tom koraku $r_i = \langle r_0^i, \ldots, r_{e+k}^i\rangle$.
Općenito, rad RAM-stroja sa programom P u spremniku programa, te sa $\vec{x}$ kao ulaznim podacima nazivamo P-**izračunavanje sa** $\vec{x}$. Za dokaz sljedeće propozicije valja pogledati [2].

Propozicija 2.5.3. *Neka je* $k \geq 1$ *prirodan broj. Neka je* $\vec{x} \in \mathbb{N}^k$, *te P program za RAM-stroj s k ulaznih podataka. Neka je e njegov indeks. Tada postoji primitivno rekurzivna relacija* T_k *tako da:*
P-izračunavanje sa $\vec{x}$ *staje u n koraka i kod P-izračunavanja sa* $\vec{x}$ *je* $y = \langle r_1, \ldots, r_n\rangle$ *ako i samo ako vrijedi* $T_k(e, \vec{x}, y)$.

Prije nego što navedemo ostatak koji je potreban za skicu dokaza Kleenijevog teorema, navest ćemo dva primjera kodiranja, kako bi i vidjeli kako se kodira program za RAM-stroj, a time i sam rad RAM-stroja.

Primjer 2.5.4. *Odredimo domenu* $Dom(\{e\}_1)$, *te* $\{e\}_1(2014)$, *gdje je* $e = 2^{129} \cdot 3^{22501} \cdot 5^{73} \cdot 7^{129}$.

Imamo dakle, da je e kod niza $(128, 22500, 72, 128)$. *Želimo vidjeti kojeg je programa to kod. Prvi član niza je* 128, *te njega možemo zapisati kao* $128 = 2^7 = 2^{6+1}$. *Broj* 6 *je dakle kod prve instrukcije RAM-programa. Pošto je* $6 = 2^{1+0} \cdot 3^{1+0}$, *tj.* $6 = \langle 0, 0\rangle$, *tada je* 6 *kod instrukcije INC* $\mathcal{R}_0$. *Drugi član niza je* 22500. *Njega zapisujemo kao* $22500 = 2^{1+1} \cdot 3^{1+1} \cdot 5^{3+1}$, *a to je kod od* $\langle 1, 1, 3\rangle$. *Tada je to kod instrukcije DEC* $\mathcal{R}_1, 3$. *Analogno dobivamo da treći član niza predstavlja instrukciju GO TO* 1, *te četvrti INC* $\mathcal{R}_0$. *Dakle, program čiji je e kod je:*

1. $INC\ \mathcal{R}_0$
2. $DEC\ \mathcal{R}_1, 3$
3. $GO\ TO\ 1$
4. $INC\ \mathcal{R}_0$

Konačno, imamo da je $Dom(\{e_1\}) = \mathbb{N}$, *te je* $\{e\}_1(2014) = 2016$

Primjer 2.5.5. *Izračunajmo* $Dom(\{e\}_2)$, *te* $\{e\}_1(17, 19)$, *gdje je* $e = 2^{1501} \cdot 3^{73} \cdot 5^{22501} \cdot 7^{19}$.

Po istom principu računanja instrukcija iz prethodnog primjera, dobivamo sljedeći program čiji je e kod:

1. $DEC\ \mathcal{R}_0, 2$
2. $GO\ TO\ 1$
3. $DEC\ \mathcal{R}_1, 3$
4. $INC\ \mathcal{R}_1$

Vidimo da u 1. *i* 2. *instrukciji programa imamo beskonačnu petlju za bilo koji ulazni podatak. Zaključujemo da funkcija* $\{e_2\}$ *nije definirana niti za jedan prirodan broj, tj.* $Dom(\{e\}_2) = \emptyset$, *te* $\{e\}_1(17, 19)$ *nije definirano.*

Kako bi se dokazao Kleenijev teorem o normalnoj formi još se pokaže ta postoji primitivno rekurzivna funkcija φ takva da za svaki $k \in \mathbb{N}$ i svaki $\vec{x} \in \mathbb{N}^k$ vrijedi $\varphi(\langle \vec{x}\rangle) = x_k$. Tada se definira $U(y) = (\varphi(y))_1$, a za indeks funkcije f se uzme kod e nekog programa P koji izračunava funkciju f. Postojanje takvog programa slijedi iz propozicije 1.3.4. Kleenijev teorem ima mnoge posljedice, koje su navedene i dokazane u [2]. Ovdje ćemo iskazati i dokazati jednu od njih. Da bi to mogli, prvo ćemo dati jednu definiciju, te iskaz jednog teorema. Dokaz se može vidjeti u [2].

Definicija 2.5.6. *Neka je* $\varphi : S \subseteq \mathbb{N}^k \to \mathbb{N}$ *neka funkcija. Kažemo da za funkciju* φ *postoji* ***indeks*** *ako postoji* $e \in \mathbb{N}$ *takav da za sve* $\vec{x} \in \mathbb{N}^k$ *vrijedi* $\varphi \simeq \{e\}^k(\vec{x})$.

Teorem 2.5.7. *Funkcija $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$ je parcijalno rekurzivna ako i samo ako postoji indeks za f.*

Teorem 2.5.8. *Funkcija je RAM-izračunljiva ako i samo ako je parcijalno rekurzivna.*

Dokaz. Ako je funkcija f parcijalno rekurzivna tada iz propozicije 1.3.4 sljiedi da je i RAM-izračunljiva.
Ako je funkcija f RAM-izračunljiva, tada iz definicije slijedi da postoji neki program P za RAM-stroj koji ju izračunava. Neka je e kod programa P. Iz dokaza Kleenijevog teorema tada slijedi da je e jedan indeks funkcije f. Iz prethodnog teorema očito slijedi da je f parcijalno rekurzivna. □

Za parcijalno rekurzivne funkcionale vrijedi analogon Kleenijevog teorema o normalnoj formi za parcijalno rekurzivne funkcije. Prije iskaza i dokaza teorema ćemo dati definicije nekih rekurzivnih funkcija i relacija koje ćemo koristiti u dokazu te definirat konačnu aproksimaciju funkcije.

Neke rekurzivne funkcije koje ćemo koristiti u dokazu:

- jednomjesna totalna funkcija $ln : \mathbb{N} \to \mathbb{N}$, gdje je $ln(x)$ funkcija koja vraća duljinu koda konačnog niza prirodnih brojeva.
- funkcija $exp : \mathbb{N}^2 \to \mathbb{N}$, gdje je sa $exp(x,i)$ označen eksponent prostog broja p_i u rastavu broja x na proste faktore.
- jednomjesna relacija Seq koja je definirana sa: $Seq(x)$ ako i samo ako x je kod nekog konačnog niza prirodnih brojeva.

Definicija 2.5.9. *Neka je $\mathcal{T}$ zadana rekurzivna relacija. Za sve $e,s,n \in \mathbb{N}$ definiramo funkciju $\{e\}^n_s$ ovako:*

$$\{e\}^n_s(\vec{x}) \simeq \begin{cases} \{e\}^n(\vec{x}), & ako\ (\exists y < s)\mathcal{T}_n(e,\vec{x},y) \\ nedefinirano, & inače \end{cases}$$

Funkciju $\{e\}^n_s$ nazivamo ***konačnom aproksimacijom*** *funkcije $\{e\}^n$.*

Teorem 2.5.10 (O normalnoj formi za parcijalno rekurzivne funkcionale)**.** *Postoji primitivno rekurzivna funkcija U i za sve $m,n \in \mathbb{N}$ postoji rekurzivna relacija na funkcijama i brojevima $T_{m,n}$ takva da za svaki parcijalno rekurzivni funkcional F s m funkcijskih i n brojevnih varijabli postoji prirodan broj $e \in \mathbb{N}$ takav da vrijedi:*

1. $F(f_1,\ldots,f_m,x_1,\ldots,x_n)\downarrow$ *ako i samo ako* $\exists y\mathcal{T}_{m,n}(e,f_1,\ldots,f_m,x_1,\ldots,x_n,y)$

2. $F(f_1,\ldots,f_m,x_1,\ldots,x_n)\simeq U(\mu y\mathcal{T}_{m.n}(e,f_1,\ldots,f_m,x_1,\ldots,x_n,y))$

Broj e zovemo ***indeks*** *funkcionala F.*

Dokaz. Vidimo da funkcije $f_1,\ldots,f_m$ ne moraju biti rekurzivne, pa ćemo koristiti njihove konačne aproksimacije. Ideja dokaza je pridružiti prirodne brojeve (kodove) funkcionalima i izračunavanjima na način da relacija na funkcijama i brojevima $T_{m.n}$ zadovoljava sljedeće: y je broj pridružen izračunavanju vrijednosti funkcionala s indeksom e i ulaznim funkcijama $f_1,\ldots,f_m$ te ulaznim parametrima $x_1,\ldots,x_n$. Iz ovoga slijedi da je $\mu yT_{m.n}(e,f_1,\ldots,f_m,x_1,\ldots,x_n,y)$ kod koji predstavlja jedno takvo izračunavanje, a funkcija U će tada dati izlazni rezultat. Dokaz provodimo u nekoliko koraka:

1. Rekurzivnim funkcijama i funkcionalima pridružujemo prirodne brojeve, tj. indekse.

- $\langle 0\rangle\mapsto\mathcal{O}$, gdje je $\mathcal{O}$ nul-funkcija;
- $\langle 1\rangle\mapsto\mathcal{S}$, gdje je $\mathcal{S}$ funkcija sljedbenika;
- $\langle 2,n,i\rangle\mapsto\mathcal{I}_i^n$ za $1\leq i\leq n$, gdje je $\mathcal{I}_i^n$ projekcija;
- $\langle 3,b_1,\ldots,b_m,a\rangle\mapsto F(f_1,\ldots,f_m,\vec{x})=G(H_1(f_1,\ldots,f_m,\vec{x}),\ldots,H_m(f_1,\ldots,f_m,\vec{x}))$, gdje su $b_1,\ldots,b_m,a$ redom indeksi parcijalno rekurzivnih funkcionala $H_1,\ldots,H_m$ i G;
- $\langle 4,a,b\rangle\mapsto F(f_1,\ldots,f_m,\vec{x},y)$, gdje je F funkcional dobiven pomoću primitivne rekurzije iz funkcionala G i H i gdje su a i b redom indeksi funkcionala G i H;
- $\langle 5,a\rangle\mapsto F(f_1,\ldots,f_m,\vec{x})=\mu y(G(f_1,\ldots,f_m,\vec{x},y)=0$, ako $\forall x\exists y(G(f_1,\ldots,f_m,\vec{x},y)=0)$, gdje je a indeks funkcionala G.

2. Zapisat ćemo izračunavanje u kanonskom obliku.

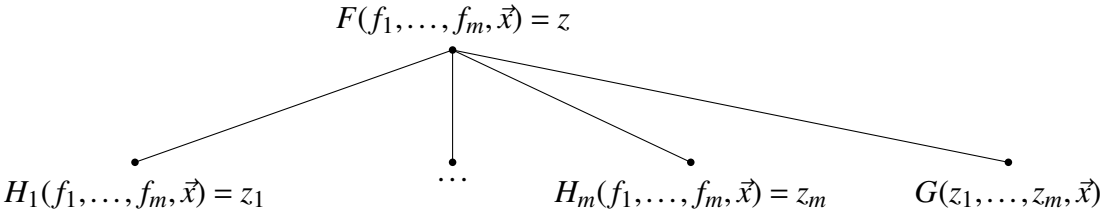
Proces izračunavanja vrijednosti za neki dani rekurzivni funkcional pregledno zapisujemo pomoću takozvanog **stabla izračunavanja**. Svaki čvor takvog stabla nam govori kako rekurzivno dobiti neku vrijednost koja se pojavljuje u izračunavanju. Definiramo stabla izračunavanja rekurzivnog funkcionala te ukazujemo na broj prethodnika kojeg svaki čvor ima.

• za sljedeće funkcionale stablo izračunavanja se sastoji od samo jednog čvora:

(*i*) nul-funkcija,
(*ii*) funkcija sljedbenika, te
(*iii*) projekcija.

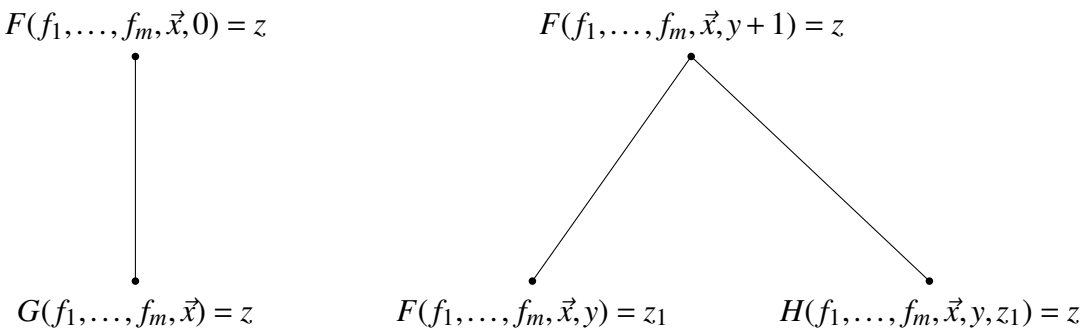
• kompozicija

Ako je $F(f_1,\ldots,f_m,\vec{x}) = G(H_1(f_1,\ldots,f_m,\vec{x},y),\ldots,H_m(f_1,\ldots,f_m,\vec{x},y))$ tada čvor $F(f_1,\ldots,f_m,\vec{x}) = z$ ima $m+1$ prethodnika:



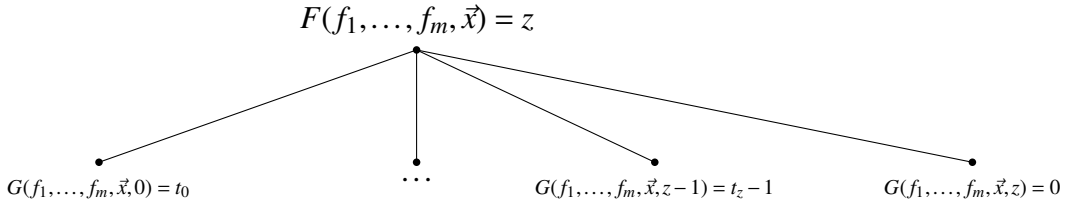
• primitivna rekurzija

Ako je funkcional $F(f_1,\ldots,f_m,\vec{x},y)$ definiran pomoću primitivne rekurzije iz funkcionala G i H imamo dva slučaja (redom s jednim, odnosno dva prethodnika):



• μ-operator

Ako je $F(f_1,\ldots,f_m,\vec{x}) = \mu y(G(f_1,\ldots,g_m,\vec{x},y) = 0)$ onda nemamo fiksan način određivanja prethodnika od $F(f_1,\ldots,f_m,\vec{x}) = y$ pa promatramo općenitu situaciju:



3. Stablima izračunavanja pridružujemo prirodne brojeve.

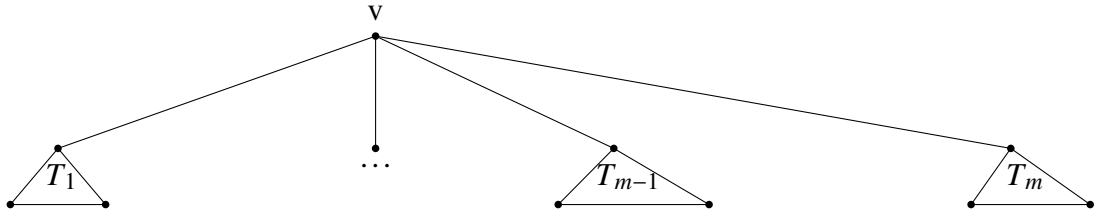
Ovaj korak ćemo napraviti indukcijom po konstrukciji stabla izračunavanja. Prvo pridružujemo čvorovima prirodne brojeve. Kako čvorovi predstavljaju izraze oblika $F(f_1,\ldots,f_m, \vec{x}) = z$, pridružujemo im sljedeće kodove:

$$\langle e, \langle x_1,\ldots,x_n\rangle, \langle g_1,\ldots,g_m\rangle, z\rangle$$

gdje je e indeks funkcionala, a g_i predstavljaju konačne aproksimacije ulaznih funkcija f_i.

Vidimo da je čvor stabla predstavljen s indeksom funkcionala, kodom ulaznih funkcija i ulaznih parametara te izlaznim rezultatom.

Sada ćemo stablima izračunavanja također pridružiti prirodne brojeve na sljedeći način: svako stablo T sastoji se od korijena v kojemu pridružujemo broj v te od konačnog broja sljedbenika (uključujući i nula sljedbenika), od kojih svaki predstavlja jedno podstablo T_i stabla T.



Rekurzivno ovom stablu pridružujemo sljedeći kod

$$\widehat{T} = \langle v, \widehat{T_1}, \ldots, \widehat{T_m} \rangle,$$

gdje je $\widehat{T_i}$ kod pridružen podstablu T_i. Posebno, ako korijen kojemu je pridružen broj v nema prethodnika, onda je sam stablo s pridruženim kodom $\langle v \rangle$.

4. Činjenicu da je y kod nekog stabla izračunavanja zapisat ćemo kao rekurzivnu relaciju $\mathcal{T}(y)$.

Radi preglednijeg zapisa, u nastavku ćemo ispuštati zagrade te ih zamjeniti zarezima. Na primjer, umjesto $(((a)_i)_j)_k$ pisat ćemo $(a)_{i,j,k}$. Znamo da je

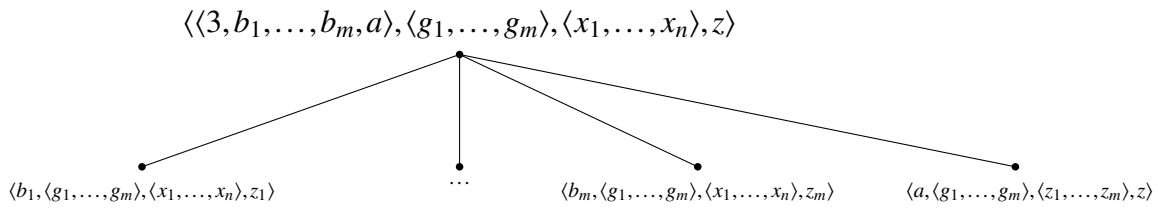
$$y = \langle v, \widehat{T_1}, \ldots, \widehat{T_m} \rangle,$$

stoga

$$\begin{aligned}
(y)_1 &= \langle e, \langle g_1, \ldots, g_m \rangle, \langle x_1, \ldots, x_n \rangle, z \rangle \\
(y)_{1,1} &= e \\
(y)_{1,2} &= \langle x_1, \ldots, x_n \rangle \\
(y)_{1,3} &= \langle g_1, \ldots, g_m \rangle \\
(y)_{1,4} &= z \\
(y)_{i+1} &= \widehat{T_i} \\
(y)_{i+1,1} &= \text{broj pridružen korijenu stabla } T_i.
\end{aligned}$$

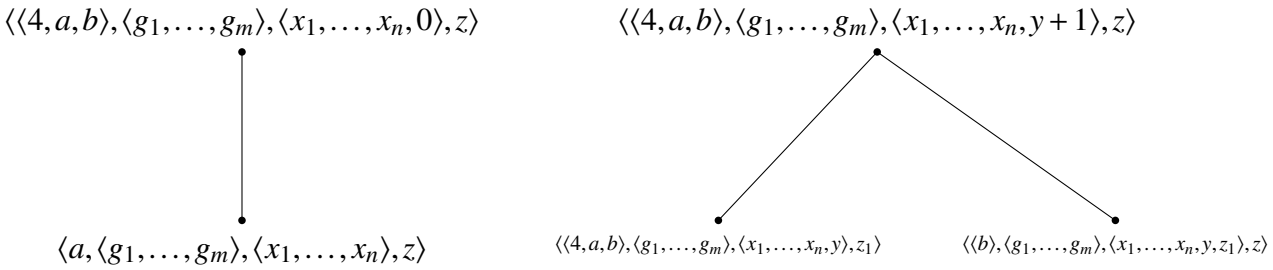
Potom iz koraka 2. imamo četiri različita slučaja definicije relacije na funkcijama i brojevima $\mathcal{T}$ koja navodimo redom skupa s pripadnom skicom stabla izračunavanja:

a) kompozicija



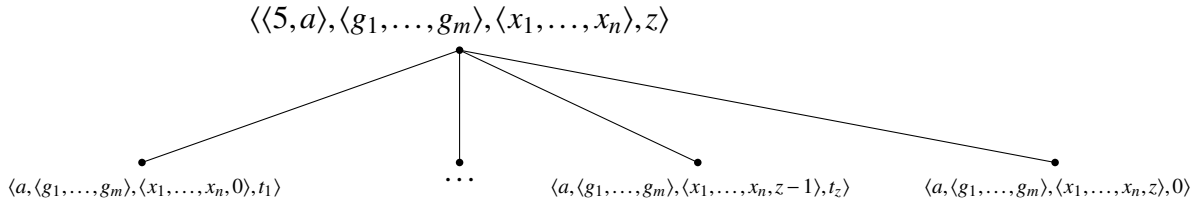
Iz gornjeg stabla imamo: $\langle b_1, \langle g_1, \ldots, g_m \rangle, \langle x_1, \ldots, x_n \rangle, z_1 \rangle = (y)_{2,1}$, $\langle b_m, \langle g_1, \ldots, g_m \rangle, \langle x_1, \ldots, x_n \rangle, z_m \rangle = (y)_{m+1,1}$, $\langle a, \langle g_1, \ldots, g_m \rangle, \langle z_1, \ldots, z_m \rangle, z \rangle = (y)_{m+2,1}$, gdje su $(y)_{2,1}$, $(y)_{m+1,1}$ te $(y)_{m+2,1}$ brojevi pridruženi korijenu $\langle \langle 3, b_1, \ldots, b_m, a \rangle, \langle g_1, \ldots, g_m \rangle, \langle x_1, \ldots, x_n \rangle, z \rangle$ stabla izračunavanja za kompoziciju.

b) primitivna rekurzija (dva slučaja):



Iz stabla za primitivnu rekurziju je $\langle a, \langle g_1, \ldots, g_m \rangle, \langle x_1, \ldots, x_n \rangle, z \rangle = (y)_{2,1}$, $\langle \langle 4, a, b \rangle, \langle g_1, \ldots, g_m \rangle, \langle x_1, \ldots, x_n, y + 1 \rangle, z \rangle = (y)_{2,1}$ te $\langle \langle b \rangle, \langle g_1, \ldots, g_m \rangle, \langle x_1, \ldots, x_n, y, z_1 \rangle, z \rangle = (y)_{3,1}$, gdje je $(y)_{2,1}$ broj pridružen korijenu $\langle \langle 4, a, b \rangle, \langle g_1, \ldots, g_m \rangle, \langle x_1, \ldots, x_n, 0 \rangle, z \rangle$ stabla izračunavanja za za prvi slučaj primitivne rekurzije te $(y)_{2,1}$ i $(y)_{3,1}$ brojevi pridruženi korijenu $\langle \langle 4, a, b \rangle, \langle g_1, \ldots, g_m \rangle, \langle x_1, \ldots, x_n, y + 1 \rangle, z \rangle$ stabla izračunavanja za drugi slučaj primitivne rekurzije.

c) μ-operator



Iz stabla za μ-operator imamo da je $\langle a, \langle g_1, \ldots, g_m\rangle, \langle x_1, \ldots, x_n, 0\rangle, t_1\rangle = (y)_{2,1}$, $\langle a, \langle g_1, \ldots, g_m\rangle, \langle x_1, \ldots, x_n, z-1\rangle, t_z\rangle = (y)_{z+1,1}$ te $\langle a, \langle g_1, \ldots, g_m\rangle, \langle x_1, \ldots, x_n, z\rangle, 0\rangle = (y)_{z+2,1}$, gdje su $(y)_{2,1}$, $(y)_{z+1,1}$ te $(y)_{z+2,2}$ brojevi pridruženi korijenu $\langle\langle 5, a\rangle, \langle g_1, \ldots, g_m\rangle, \langle x_1, \ldots, x_n\rangle, z\rangle$ stabla izračunavanja za μ-operator.

Sada ćemo definirati jednomjesnu relaciju A ovako:

$$A(y) \iff Seq(y) \wedge Seq((y)_1) \wedge ln((y)_1) = 3 \wedge Seq((y)_{1,1}) \wedge Seq((y)_{1,2}).$$

Očito je relacija A rekurzivna. Lako je vidjeti da za svaki prirodan broj $y \in \mathbb{N}$ vrijedi: $A(y)$ ako i samo ako y je kod nekog čvora stabla izračunavanja.

Definiramo jednomjesnu relaciju B ovako:

$$\begin{aligned} B(y) \iff\ & ln(y) = 1 \wedge \\ & \{[(y)_{1,1} = \langle 0\rangle \wedge ln((y)_{1,2}) = 1 \wedge (y)_{1,3} = 0] \vee \\ & [(y)_{1,1} = \langle 1\rangle \wedge ln((y)_{1,2}) = 1 \wedge (y)_{1,3} = (y)_{1,2,1} + 1] \vee \\ & [ln((y)_{1,1}) = 3 \wedge (y)_{1,1,1} = 2 \wedge (y)_{1,1,2} = ln((y)_{1,2}) \wedge \\ & 1 \leq (y)_{1,1,3} \leq (y)_{1,1,2} \wedge (y)_{1,3} = ((y_{1,2}))_{(y)_{1,1,3}}]\}. \end{aligned}$$

Očito je relacija B rekurzivna. Lako je vidjeti da za svaki prirodan broj $y \in \mathbb{N}$ vrijedi: $B(y)$ ako i samo ako y je indeks neke od inicijalnih funkcija.

Definiramo jednomjesnu relaciju C ovako:

$$\begin{aligned} C(y) \iff\ & ln((y)_{1,1}) \geq 3 \wedge (y)_{1,1,1} = 3 \wedge ln(y) = ln((y)_{1,1}) \wedge \\ & (\forall i)_{2 \leq i < ln(y)}[(y)_{i,1,1} = (y)_{1,1,i} \wedge (y)_{i,1,2} = (y)_{1,2}] \wedge \\ & (y)_{ln(y),1,1} = (y)_{1,1,ln(y)} \wedge (y)_{ln(y),1,2} \wedge \\ & (y)_{ln(y),1,2} = \langle (y)_{2,1,3}, \ldots, (y)_{ln(y)-1,1,3}\rangle. \end{aligned}$$

Očito je relacija C rekurzivna. Lako je vidjeti da za svaki prirodan broj $y \in \mathbb{N}$ vrijedi: $C(y)$ ako i samo ako y je indeks funkcije koja je dobivena pomoću kompozicije funkcija.

Za primitivnu rekurziju rekli smo da imamo dva slučaja. Definiramo relaciju D ovako:

$$\begin{aligned} D(y) \iff & ln((y)_{1,1}) = 3 \wedge (y)_{1,1,1} = 4 \wedge \\ & \{[(y)_{1,2,ln((y)_{1,2})} = 0 \wedge ln(y) = 2 \wedge (y)_{2,1,1} = (y)_{1,1,2} \wedge \\ & (y)_{2,1,2} * \langle 0 \rangle = (y)_{1,2} \wedge (y)_{2,1,3} = (y)_{1,3}] \vee \\ & [(y)_{1,2,ln((y)_{1,2})} > 0 \wedge ln(y) = 3 \wedge \\ & (y)_{2,1,1} = (y)_{1,1} \wedge ln((y)_{2,1,2}) = ln((y)_{1,2}) \wedge \\ & (\forall i)_{1 \leq i < ln((y)_{1,2})}(y_{2,1,2,i} = (y)_{1,2,i}) \wedge \\ & (y)_{2,1,2,ln((y)_{1,2})} + 1 = (y)_{1,2,ln((y)_{1,2})} \wedge \\ & (y)_{3,1,1} = \langle (y)_{1,1,3} \rangle \wedge (y)_{3,1,3} = (y)_{1,3} \wedge \\ & (y)_{3,1,2} = (y)_{2,1,2} * \langle (y)_{2,1,3} \rangle]\}. \end{aligned}$$

Očito je relacija D rekurzivna te za svaki prirodan broj y vrijedi: $D(y)$ ako i samo ako y je indeks funkcije koja je dobivena pomoću primitivne rekurzije.

Definirajmo relaciju E ovako:

$$\begin{aligned} E(y) \iff & ln((y)_{1,1}) = 2 \wedge (y)_{1,1,1} = 5 \wedge \\ & ln(y) \geq 2 \wedge (y)_{1,3} = ln(y) - 2 \wedge \\ & (\forall i)_{2 \leq i \leq ln(y)}[(y)_{i,1,1} = (y)_{1,1,2} \wedge \\ & (y)_{i,1,2} = (y)_{1,2} * \langle i - 2 \rangle] \wedge \\ & (\forall i)_{2 \leq i < ln(y)}[(y)_{i,1,3} \neq 0] \wedge (y)_{ln(y),1,3} = 0. \end{aligned}$$

Očito je i relacija E rekurzivna te za svaki prirodan broj y vrijedi: $E(y)$ ako i samo ako y je indeks funkcije koja je dobivena pomoću μ-operatora.

Time smo pokrili sve moguće slučajeve te stoga možemo definirati relaciju $\mathcal{T}$ ovako:

$$\begin{aligned} \mathcal{T}(y) \iff & A(y) \wedge [B(y) \vee C(y) \vee D(y) \vee E(y)] \wedge \\ & [ln(y) > 1 \rightarrow (\forall i)_{2 \leq i \leq ln(y)} \mathcal{T}_i)]. \end{aligned}$$

Iz ovoga očito slijedi da je $\mathcal{T}$ rekurzivna relacija jer je definirana isključivo pomoću rekurzivnih izraza i vrijednosti prethodnih argumenata. Lako je vidjeti da za svaki prirodan broj y vrijedi: $\mathcal{T}(y)$ ako i samo ako y je kod nekog stabla izračunavanja.

5. Definirajmo na kraju relaciju $\mathcal{T}_{m,n}$ i funkciju $\mathcal{U}$.

$\mathcal{T}_{m,n}$ je relacija na funkcijama i brojevima koju definiramo na sljedeći način:

$$\mathcal{T}_{m,n}(e, x_1, \ldots, x_n, f_1, \ldots, f_m, y) \iff \mathcal{T}(y) \wedge$$

$$(y)_{1,1} = e \wedge (y)_{1,2} = \langle x_1, \ldots, x_n \rangle \wedge$$
$$ln((y)_{1,3}) = m \wedge (\forall i)_{1 \leq i \leq m}(f_i | (y)_{1,3,i}).$$

Očito je relacija $\mathcal{T}_{m,n}$ rekurzivna te za svaki prirodan broj y vrijedi: relacija $\mathcal{T}_{m,n}(e, x_1, \ldots, x_n, f_1, \ldots, f_m, y)$ ako i samo ako y je kod nekog stabla izračunavanja kojemu je $(y)_{1,1}$ prvi parametar koji predstavlja kod funkcije čiji je indeks e i $(y)_{1,2}$ je drugi parametar koji predstavlja kod ulaznih parametara $x_1, \ldots, x_n$ te $(y)_{1,3}$ treći parametar koji predstavlja kod ulaznih funkcija $f_1, \ldots, f_m$ duljine m.

Konačno, kako su čvorovi uređene četvorke te je izlazni rezultat njihov zadnji parametar, definiramo $\mathcal{U}$ kao

$$\mathcal{U}(y) = (y)_{1,4}.$$

□

Za slučaj ograničenih funkcionala može se pokazati jači rezultat.

Teorem 2.5.11 (O normalnoj formi za ograničene parcijalno rekurzivne funkcionale)**.** *Postoji primitivno rekurzivna funkcija U i za sve $m, n \in \mathbb{N}$ postoji primitivno rekurzivna relacija $\mathcal{T}_{m,n}$ takva da za svaki parcijalno rekurzivni ograničen funkcional F s m funkcijskih i n brojevnih varijabli postoji prirodni broj e tako da vrijedi:*

1. $F(f_1, \ldots, f_m, x_1, \ldots, x_n) \downarrow$ *ako i samo ako* $\exists y \mathcal{T}_{m,n}(e, \tilde{f}_1(y), \ldots, \tilde{f}_m(y), x_1, \ldots, x_n, y)$

2. $F(f_1, \ldots, f_m, x_1, \ldots, x_n) \simeq U(\mu y \mathcal{T}_{m.n}(e, \tilde{f}_1(y), \ldots, \tilde{f}_m(y), x_1, \ldots, x_n, y))$

Dokaz. Dovoljno je iz prethodnog dokaza modificirati definiciju relacije $\mathcal{T}_{m,n}$ ovako:

$$\mathcal{T}_{m,n}(e, x_1, \ldots, x_n, f_1, \ldots, f_m, y) \iff \mathcal{T}(y) \wedge$$
$$(y)_{1,1} = e \wedge (y)_{1,2} = \langle x_1, \ldots, x_n \rangle \wedge$$
$$ln((y)_{1,3}) = m \wedge (\forall i)_{1 \leq i \leq m}(Seq(z_i) \wedge$$
$$ln(z_i) = y + 1 \wedge z_i | (y)_{1,3,i}).$$

Oznaka $z|g$ znači da z proširuje funkciju čiji je kod g. Ideja je da moramo uzeti izračunavanje čiji je kod y uzimajući u obzir konačne funkcije koje se pojavljuju u njemu (čiji su kodovi $(y)_{1,3,i} < y$) te primijetiti da se u izračunavanju mogu koristiti samo argumenti funkcija f_i koji su manji ili jednaki od y. Sve te vrijednosti su kodirane ulaznim vrijednostima $\tilde{f}_i(y)$ relacije $\mathcal{T}_{m,n}$. Ovakva definicija relacije $\mathcal{T}_{m,n}$ je složenija zbog činjenice da se funkcije f_i ne javljaju direktno u njoj, već preko vrijednosti ulaznih funkcija $\tilde{f}_i(y)$, čiju ulogu su preuzeli z_i.

□

Sljedeći cilj nam je dokazati prvi teorem rekurzije za parcijalno rekurzivne funkcionale. Kako bi to mogli, prvo ćemo uvesti pojam fiksne točke te iskazati i dokazati jednu propoziciju.

2.6 Pojam fiksne točke

Proizvoljnu funkciju α možemo definirati na način da definiramo uvjete koje vrijednosti te funkcije moraju zadovoljavati. Ako ti uvjeti sadrže samu funkciju α, tada njena definicija poprima općeniti oblik

$$\alpha(x) \simeq F(\alpha, x),$$

za neki funkcional F (gdje F ne mora biti rekurzivan). Svaka funkcija koja zadovoljava gornji izraz zove se **fiksna točka** funkcionala F te se može smatrati definiranom preko danih uvjeta. No svrha definicije nije samo specificirati i dati sve potrebne informacije, nego i izbaciti dodatne, ne eksplicitno navedene informacije. Stoga možemo uzeti u obzir funkciju definiranu s danim uvjetima ako je ta funkcija **najmanja fiksna točka** funkcionala F: u tom slučaju funkcija neće sadržavati proizvoljne informacije, što na kraju i vidimo iz definicije funkcije α.

Primjer 2.6.1. *Neka je F funkcional definiran sa $F(\alpha, x) \simeq \alpha(x)$. Svaka parcijalna funkcija je fiksna točka funkcionala F, a najmanja fiksna točka je stoga funkcija koja nije definirana niti za jedan prirodan broj. Konkretno, F je totalan funkcional s totalnim fiksnim točkama, no najmanja fiksna točka nije totalna.*

Primjer 2.6.2. *Neka je F funkcional definiran s $F(\alpha, x) \simeq \alpha(x) + 1$. U ovom slučaju imamo točno jednu fiksnu točku, a to je funkcija koja nije definirana niti za jedan prirodan broj. Dakle, F je totalan funkcional koji ne sadrži totalne fiksne točke.*

Primjer 2.6.3. *Neka je R rekurzivna relacija za koju vrijedi $R(x, y)$, gdje su $x, y \in \mathbb{N}$ zadani. Neka je F funkcional definiran ovako:*

$$F(\alpha, x, y) \simeq \begin{cases} y, & \text{ako vrijedi } R(x, y) \\ \alpha(x, y+1), & \text{inače.} \end{cases}$$

Tada je F parcijalno rekurzivan funkcional te ako je α najmanja fiksna točka, tada je α parcijalno rekurzivna funkcija i

$$\alpha(x,y) \simeq \mu z(z \geq y \wedge R(x,z)).$$

Posebno imamo $\alpha(x,0) \simeq \mu y R(x,y)$.

2.7 Prvi teorem rekurzije

Kako bi mogli dokazati prvi teorem rekurzije još nam je ostalo iskazati i dokazati jednu propoziciju.

Propozicija 2.7.1. *Neka je* $F(f,x)$ *parcijalno rekurzivan funkcional,* f *neka funkcija,* $x \in \mathbb{N}$ *te* $y \simeq F(f,x)$. *Tada:*

1. *Postoji restrikcija u funkcije* f *s konačnom domenom takva da vrijedi* $F(u,x) \simeq y$. ***(kompaktnost)***

2. *Ako je* g *proširenje funkcije* f *tada je* $F(g,x) \simeq y$. ***(monotonost)***

Dokaz. Prema definiciji relacije $\mathcal{T}_{m,n}$ iz teorema 2.5.10, vrijednost funkcionala F se izračunava koristeći konačnu aproksimaciju funkcije f. Iz toga odmah slijedi kompaktnost. Monotonost slijedi iz činjenice da je svaka konačna aproksimacija funkcije f ujedno i konačna aproksimacija svake funkcije koja proširuje funkciju f.

□

Teorem 2.7.2 (Prvi teorem rekurzije)**.** *Za svaki parcijalno rekurzivni funkcional* $F(f,x)$ *postoji najmanja parcijalno rekurzivna fiksna točka* f, *tj. postoji parcijalno rekurzivna funkcija* f *za koju vrijedi:*

1. $f(x) \simeq F(f,x)$.

2. *Ako za neku funkciju* g *vrijedi* $g(x) \simeq F(g,x)$ *tada je* g *proširenje funkcije* f.

Dokaz. Definirajmo rekurzivno niz funkcija (f_n) ovako::

(i) f_0 je funkcija koja nije definirana ni u jednoj točki.

(ii) $f_{n+1}(x) \simeq F(f_n,x)$.

Intuitivno, prvo uzimamo sve vrijednosti funkcionala F koje se mogu izračunati bez poziva na funkciju f. Zatim ćemo, u bilo kojoj danoj fazi izračunavanja računati sve one

vrijednosti funkcionala F koje koriste pozive funkcije f iz razloga što su već izračunate u prethodnim fazama.

Indukcijom po n pokazat ćemo da je $f_n \subseteq f_{n+1}$.

- Za $n = 0$ tvrdnja vrijedi jer je funkcija f_0 nedefinarana u bilo kojoj točki.
- Ako je $f_n \subseteq f_{n+1}$ neka je $f_{n+1}(x) \simeq F(f_n, x) \simeq y$. Prema svojstvu monotonosti iz propozicije 2.7.1 imamo $F(f_{n+1}, x) \simeq y$. Tada je $f_{n+2}(x) \simeq y$ te $f_{n+1} \subseteq f_{n+2}$.

Stoga ima smisla uzeti u obzir ograničenje f funkcija f_n. Odnosno, definiramo funkciju f ovako:

$$f(x) \simeq y \iff \exists n(f_n(x) \simeq y).$$

Funkcija f je parcijalno rekurzivna jer su funkcije f_n parcijalno rekurzivne.

Štoviše:

1. Funkcija f je fiksna točka funkcionala F.

Ako vrijedi $f(x)\downarrow$ tada za neke n vrijedi sljedeće:

$$f(x) \simeq f_{n+1}(x) \simeq F(f_n, x).$$

Kako je $f_n \subseteq f$, iz svojstva monotonosti prema propoziciji 2.7.1 slijedi $f(x) \simeq F(f, x)$.

Ako vrijedi $F(f, x)\downarrow$, onda se prema svojstvu kompaktnosti iz propozicije 2.7.1 u izračunavanju koristi samo konačno mnogo vrijednosti funkcije f te će stoga f_n, za dovoljno veliki n biti dovoljan. Tada je

$$F(f, x) \simeq F(f_n, x) \simeq f_{n+1}(x) \simeq f(x).$$

2. Funkcija f je najmanja fiksna točka funkcionala F.

Pretpostavimo da je $F(g, x) \simeq g(x)$, za sve $x \in \mathbb{N}$. Indukcijom po n imamo $f_n \subseteq g$, te je stoga $f \subseteq g$:

- $f_0 \subseteq g$ jer je funkcija f_0 nedefinirana u svakoj točki.
- ako je $f_n \subseteq g$ tada prema svojstvu monotonosti iz 2.7.1, slijedi

  $$f_{n+1}(x) \simeq F(f_n, x) \simeq F(g, x) \simeq g(x),$$

  te stoga $f \subseteq g$.

□

Bibliografija

[1] P. Odifreddi, *Classical recursion theory*, Elsevier, 1992.

[2] M. Vuković, *Izračunljivost, nastavni materijal*, PMF-MO, 2009, https://www.math.pmf.unizg.hr/sites/default/files/pictures/izn-skripta-2009.pdf.

[3] Wikipedia, *Oracle machine — Wikipedia, The Free Encyclopedia*, 2016, https://en.wikipedia.org/wiki/Oracle_machine?title=Oraclemachine.

Sažetak

Cilj ovog rada bio je napraviti generalizaciju parcijalno rekurzivnih funkcija na parcijalno rekurzivne funkcionale. Rad smo započeli uvođenjem pojmova izračunavanja te izračunljive funkcije. Potom smo definirali RAM-stroj i RAM-izračunljivu funkciju kako bi mogli definirati novu klasu izračunljivih funkcija, a to su parcijalno rekurzivne funkcije. Naveli smo osnovna svojstva parcijalno rekurzivnih funkcija te dali skicu dokaza glavnog teorema za parcijalno rekurzivne funkcije, a to je Keenijev teorem o normalnoj formi. Prilikom davanja skice dokaza pokazali smo kako "efektivno kodirati" konačne nizove prirodnih brojeva definirajući funkciju kodiranja i kod konačnog niza te čitav postupak ilustrirali kroz primjere. Nakon što smo napravili uvod i definirali sve što nam je potrebno, uveli smo pojam funkcionala. Funkcional smo definirali kao funkciju koja uz prirodne brojeve može kao argumente imati i funkcije na prirodnim brojevima. Postupno generaliziramo pojmove koje smo u uvodu imali za funkcije definirajući intuitivno pojam izračunljivog funkcionala. Definiciju RAM-stroja proširujemo definicijom GRAM-stroja dodavajući novu instrukciju koja je zadužena za rad s ulaznim funkcijama nekog funkcionala. Na analogan način definiramo i GRAM-izračunljivu funkciju te definiramo da je svaka RAM-izračunljiva funkcija ujedno i GRAM-izračunljiva. Potom smo po analogonu za parcijalno rekurzivne funkcije dokazali važni teorem o normalnoj formi za parcijalno rekurzivne funkcionale. Sljedeći cilj, te ujedno i zaključak ovog rada bio je iskazati i dokazati također bitan prvi teorem rekurzije za parcijalno rekurzivne funkcionale. To smo napravali uvođenjem pojma fiksne točke te definicijom svojstva monotonosti i kompaktnosti za parcijalno rekurzivne funkcionale.

Summary

The main point of this work was to make a generalization of partial recursive functions to partial recursive functionals. First we introduced the notions of computability and a computable function. Then we defined RAM-machines and RAM-computable functions so that we could define a new class of computable functions. This is the class of partial recursive functions. We talked about the main characteristics of partial recursive functions and gave a quick review of the main theorem for partial recursive functions, which is Kleene's normal form theorem. While we gave the quick proof we also introduced the notion of "effective coding" of finite sequences of natural numbers by definig the coding function and the code of a finte sequence of natural numbers. We then gave examples of the coding procedure. After we defined everything needed, we introduced the notion of a functional and defined it as a function with parameters that can take not only natural numbers as arguments, but also functions on natural numbers. Then we are stepwise doing a generalization of notions that we had for functions by intuitively giving the definition of a computable functional. We generalize the definition of RAM-machines by defining GRAM-machines which have a new type of instruction. The job of the new instruction is to take care of the function arguments of a functional. Just like the definition of RAM-computable functions we define GRAM-computable functions and say that every RAM-computable function is also GRAM-computable. We then proved the normal form theorem for partial recursive functionals using the analogon of this theorem in terms of partial recursive functions. The next goal, and the last part of this work was to give the proof the important first recursion theorem for partial recursive functionals. We have done that by defining the fixed-point and defining the property of compactness and monothonity for recursive functionals.

Životopis

Rođen sam 26.8.1990. godine u Čapljini, gdje završavam osnovnu školu. Paralelno sam upisao i završio osnovnu glazbenu školu također u Čapljini. Po tom upisujem i završavam Opću gimnaziju u Čapljini. Tijekom osnovne i srednje škole intenzivno se bavim glazbom te sviram u glazbenim sastavima raznih tipova. Nakon srednje škole odlazim u Dubrovnik te na Sveučilištu u Dubrovniku upisujem preddiplomski studij Primjenjeno/poslovno računarstvo. Paralelno upisujem i srednju Umjetničku školu Luke Sorkočevića u Dubrovniku, instrumentalni smjer na trubi. Tijekom studija sudjelovao sam na dva sveučilišna projekta, od kojih je prvi bio projekt Gospodarske komore Dubrovnik za izradu web stranica za u tom trenutku novootvorene obrte i obrtnike. Drugi je projekt bio izrada funkcionalnog web portala za samo sveučilište. Istodobno se nastavljam profesionalno baviti glazbom svirajući u raznim sastavima, uključujući zamjene u Dubrovačkom simfonijskom orkestru. Nakon završene tri godine preddiplomskog studija u Dubrovniku, upisujem Dimplomski sveučilišni studij na Prirodoslovno-matematičkom fakultetu, Matematički odsjek Sveučilišta u Zagrebu, smjer Računarstvo i matematika. Za vrijeme diplomskog studija se zapošljavam u IT firmi DABAR informatika d.o.o. kao Java programer te radim na izradi web rješenja za poslovne sustave banaka.