MATEMATIČKE OSNOVE RAČUNARSTVA

– Žarko Mijajlović –

Teorijske osnove računarstva

Da li je računastvo matematika?

Niš 2003

MATEMATIČKE OSNOVE RAČUNARSTVA

— *apstrakt* —

Postoji nekoliko apstraktnih ali ključnih pitanja od interesa u oblasti računarstva. Ova pitanja su apstraktna utoliko što se ne odnose na konkretne programske implementacije, niti na određene aparature – računarske instalacije pomoću kojih se vrše konkretna izračunavanja. Teorije u okviru kojih se ona raspravljaju pripadaju osnovama računarstva, dok su sama pitanja zapravo matematičkog karaktera. U ovom predavanju pokušaću da predstavim nekoliko pitanja iz ove oblasti i da objasnim zašto su ona važna za teoriju i praksu računarstva:

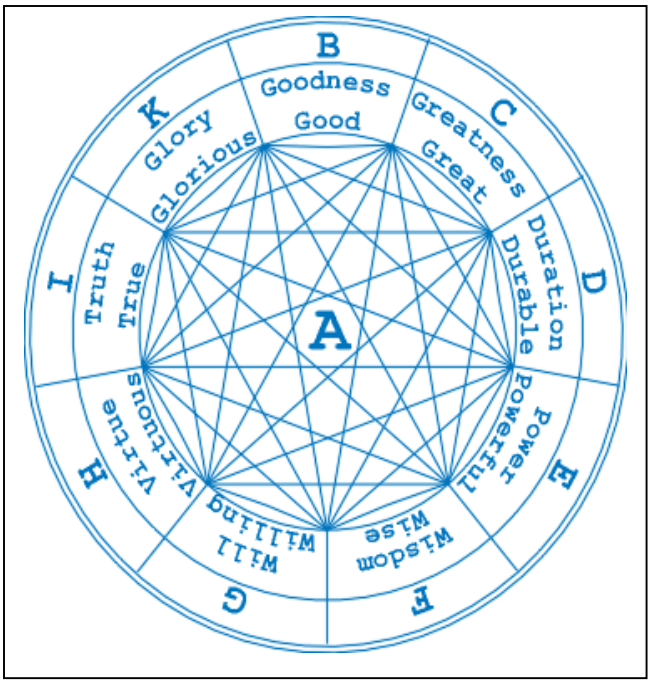
Pojam efektivne izračunljivosti. Ovaj pojam se raspravlja u okviru teorije efektivne (ili formalne) izračunljivosti. Glavno mesto ove teorije jeste pojam algoritma ili efektivnog postupka. Ova teorija daje opis izračunljivih (algoritamski rešivih) problema, ali i kriterijume pomoću kojih možemo dokazati da neki zadatak nema algoritamsko rešenje. Na primer, u okviru ove teorije dokazuje se da ne postoji efektivni postupak pomoću kojeg bi se eliminisale "beskonačne petlje" u konkretnim računarskim programima. Postoji više sistema izračunljivosti u okviru kojih se raspravlja pojam algoritma. Neki od ovih sistema su: Sistem rekurzivnih funkcija (K. Gödel), Turingove mašine (A. Turing), λ-račun (A. Church).

Složenost algoritama. Računanje prema datom algoritmu sastoji se iz konačnog niza računski elementarnih koraka. Za složenost datog algoritma možemo uzeti, na primer, dužinu ovog niza. S druge strane, jedan zadatak može se rešiti pomoću više različitih algoritama, i u tome nas zanimaju algoritmi najmanje složenosti. Najpoznatiji primer ove vrste je "Osnovni problem teorijskog računarstva: $P = NP$". Pozitivno rešenje ovog problema iznelo bi efikasnije algoritme u mnogim oblastima matematike od velikog interesa u praktičnim primenama.

Teško izračunljivi problemi. Poznati algoritmi za rešavanje određenih zadataka mogu imati veliku složenost, na primer, može biti da je potrebno eksponencijalno vreme za njegovo izvršenje u odnosu na veličinu ulaznog podatka. Ipak, za neke od ovih zadataka nije isključeno da postoje algoritmi manje složenosti (rešivost u polinomnom vremenu). Najpoznatiji primer ove vrste je 3-SAT problem (Cook). Ovaj problem je univerzalnog karaktera s obzirom da je NP-kompletan, tj. njegovo efikasno rešenje dovelo bi do efikasnijih rešenja čitave klase problema i istovremeno rešilo bi problem P=NP. Posebno su zanimljive efikasno izračunljive funkcije (problemi) za koje su poznati postupci izračunavanja inverznih funkcija veoma složeni ukoliko nisu poznati određeni parametri ("ključevi"). Tipičan zadatak ove vrste je izračunavanje proizvoda dva prirodna broja (sa većim brojem cifara, na primer 100), i njemu inverzan problem faktorizacije prirodnih brojeva. Teorija ovakvih problema našla je velike primene u teoriji kodiranja (zaštite podataka), u takozvanim kriptološkim sistemima sa javnim ključem.

Korektnost i semantika programa. Kako možemo dokazati da je neki računarski program korektan, tj. da zaista izračunava funkciju za koju je dizajniran? U tu svrhu razvijaju se formalni sistemi zasnovani na idejama teorije dokaza, u kojima je bar za neke klase programa ovaj zadatak rešiv. S druge strane dobro definisana semantika i njeno poštovanje u implementaciji, obezbeđuje istovetne rezultate izračunavanja, nezavisno od programske implementacije i programske platforme.

RAMON LULL



Po strani od glavne tradicije srednjovekovne logike nalazio se **Ramon Lull** (1232-1315) koji je smislio kombinatorni sistem elementarnih pojmova i formalni metod, nazvan ***ars universalis***. Lull je predložio svoj sistem za teološka dokazivanja hrišćanskih dogmi. Ovaj izbor elementarnih pojmova nije bio takav da bi se metoda učinila posebno plodnom. Ipak, osnovna ideja Lullieve kombinatorike imala je čara na kasnije generacije. Na primer, po svoj prilici uticala je na **Leibnizov** pokušaj da stvori univerzalni jezik nauke - ***characteristica universalis***. Lull je takođe projektovao i napravio računsku mašinu koja se sastojala od rotirajućih diskova postavljenih jedan iznad drugog. Ovom mašinom mogao se izvršavati simbolički račun iz njegovog sistema: " ***Poduhvat kojim je možda zaslužio pravo da se nazove ocem kompjuterskog programiranja***". (*Historija logike*, Zagreb, 1970, ed. A.N. Prior, glava *Srednjovjekovna logika*, Ernest A. Moody, profesor na UCLA, istaknuti stručnjak za srednjovekovnu logiku).

Ramon Lull (1232-1315) filozof, teolog, alhemista i logičar iz Katalonije. Bavio se misionarstvom u arapskim zemljama, gde je i poginuo od kamenovanja u jednoj takvoj misiji. Glavna delo: ***Ars Magna*** u kojem je izneo principe svojeg formalnog sistema i način rada svoje računske mašine ***Ars Generalis Ultima***. Delo je napisao u ime odbrane hrišćanstva, pre svega od učenja Abu-Al-Walid Muhammad Ibn-Ahmad Ibn-Rushd (1126-1198)-**Averosa**. Naime, Arabljani su imali dva standarda istine, teološki i filozofski: ono što je netačno u filozofiji (logici) može biti istinito u teologiji. Ramon je zastupao potpuno suprotno mišljenje, da nema razlike između filozofije i teologije, između rezonovanja i verovanja. Otuda se po Ramonu i najveće misterije mogu dokazati logičkim sredstvima, dakle i njegovom njegovom ***Ars Magna.***

ALGORITMI

Primeri algoritama:

1. Sabiranje i množenje prirodnih brojeva.
2. Euklidov algoritam za određivanje najvećeg zajedničkog delioca dva prirodna broja.
3. Određivanje n-te decimale broja π.
4. Diferenciranje elementarnih funkcija.
5. Rešavanje pojedinih klasa diferencijalnih jednačina.
6. Rešavanje pomoću radikala algebarskih jednačina drugog, trećeg i četvrtog stepena.
7. Rešavanje sistema linearnih jednačina (Gausov postupak).
8. Geometrijske konstrukcije uz pomoć lenjira i šestara.
9. Generisanje pseudoslučajnih brojeva.
10. Postupci za utvrđivanje tautologičnosti iskaznih formula.

Primeri matematičkih zadataka koji nisu algoritamski rešivi:

Niti za jedan od sledećih primera ne postoji univerzalan i efektivan postupak kojim bi se rešio proizvoljan zadatak iz navedene klase.

1. Rešavanje algebarskih diofantovskih jednačina (J. Matijaševič; Deseti Hilbertov problem).
2. Utvrđivanje istinitosti u strukturi prirodnih brojeva **N** aritmetičkih iskaza predstavljivih u predikatskom računu prvog reda (K. Gedel; Drugi Hilbertov problem).
3. Utvrđivanje da li se program napisan u programskom jeziku **C** za proizvoljne ulazne podatke zaustavlja posle konačno mnogo koraka (Halting problem).

Osobine koje procedure izvesnog algoritamskog sistema moraju posedovati:

1. Svaki algoritam dat je ***konačnim nizom instrukcija***.
2. Postoji ***računsko sredstvo*** koje interpretira i izvodi instrukcije algoritma.
3. Izračunavanje prema datom algoritmu je ***diskretne prirode***, dakle izvodi se korak po korak i bez korišćenja neprekidnih metoda ili analognih sredstava.
4. Postoji ***memorijski prostor*** u kojem se čuvaju, privremeno ili stalno, svi podaci koji se javljaju prilikom izračunavanja.
5. Izračunavanje prema datom algoritmu je ***determinisano***, tj. izvodi se bez korišćenja slučajnih metoda ili sredstava. Dakle, ponovljene primene algoritma na iste ulazne veličine proizvode iste izlazne veličine.
6. ***Nema ograničenja*** na veličinu ulaznih podataka, broj instrukcija, veličinu memorije, kao ni dužinu računa koji se izvodi za konkretne ulazne podatke.
7. Algoritam ***ne mora proizvesti rezultat*** za sve ulaze. Dakle, moguće je da se za određene ulazne podatke izračunavnje prema nekom algoritmu nikada ne završi.
8. U određenom smislu postoji granica u mogućnostima računskih sredstava: postoji ***univerzalan algoritam*** koji simulira izračunavanje po svakom drugom algoritmu.
9. Algoritama i objekata na kojima se algoritmi izvode ima ***prebrojivo mnogo***, ali ne i više.
10. Algoritmi, ulazni i izlazni simboli mogu se ***efektivno kodirati*** u skupu prirodnih brojeva.

Algoritamski sistem

Algoritamski sistem je formalan sistem ***S*** u okviru kojeg se matematičkim sredstvima definiše pojam ***efektivne izračunljivosti***, odnosno pojam ***algoritma***. Sistemom ***S*** definiše se skup artmetičkih funkcija $F_S = \{ f \mid f\colon \mathbf{N} \to \mathbf{N} \}$.

- Za funkcije iz F_S kažemo da su ***S***-izračunljive.
- Ako aritmetička funkcija ***f*** ne pripada F_S kažemo da ***f*** nije ***S***-izračunljiva.
- Neka je ***X*** podskup skupa prirodnih brojeva ***N***.
  Ako je karakteristična funkcija k_X skupa ***X*** ***S***-izračunljiva, kažemo da je ***X*** ***S-izračunljiv skup***.
  Ako je ***X*** skup vrednosti neke ***S***-izračunljive funkcije, onda je skup ***X*** ***S-nabrojiv*** (semiizračunljiv).
- Ako se nekom matematičkom zadatku ***M*** može nekako pridružiti ***S***-izračunljiva funkcija koja taj zadatak rešava, kažemo da je zadatak ***M algoritamski rešiv u S***. Ako to nije moguće, kažemo da je ***M algoritamski nerešiv*** u sistemu ***S***.

Pitanje 1. Da li je moguće govoriti o algoritamskoj rešivosti matematičkih zadataka koji nisu aritmetičkog karaktera?

Pitanje 2. Da li algoritamska rešivost zadatka ***M*** zavisi od izbora algoritamskog sistema, tj. da li je moguće da je ***M*** rešiv u sistemu izračunljivosti ***S***, ali ne i u ***S***'?

Pitanje 3. Da li ima neizračunljivih aritmetičkih funkcija?

Pitanje 4. Da li postoji skup ***X*** koji je ***S***-nabrojiv, ali ne i **S**-izračunljiv?

Pitanje 5. Da li postoji matemačka formulacija izračunljivosti?

1. Rešavanju prvog pitanja može se pristupiti na dva načina:

a. Da se dopusti formulacija sistema izračunljivosti na širim domenima, recimo na rečima nekog formalnog jezika.

b. Da se domen podataka D na koje se odnosi zadatak M kodira u skupu N, tj. pronadje 1–1 preslikavanje $f: D \to N$.

Prvi pristup popularniji je medu računarcima, drugi medju matematičarima koje uglavnom zanima opšta teorija izračunljivosti. Drugi naziv za kodiranje je gedelizacija prema Gedelu koji je prvi primenio ovu ideju u matematici tridesetih godina 20.v.

Primer 1. (Kantorova funkcija nabrajanja) Funkcija $f(m,n) = \binom{m+n+1}{2} + m$ kodira parove prirodnih brojeva.

Primer 2. (Gedelovo kodiranje) Ako su kodovi osnovnih simbola aritmetike i logike:

0	$\leq$	$+$	$\cdot$	$\Rightarrow$	$\wedge$	$\vee$	$\neg$	$\exists$	$\forall$	$=$	v_i
1	2	3	4	5	6	7	8	9	10	11	$20+i$

tada je kôd neke aritmetičke formule $\varphi = s_1 s_2 s_3 \dots s_k$

$$\ulcorner \varphi \urcorner = 2^{\ulcorner s_1 \urcorner} 3^{\ulcorner s_2 \urcorner} 5^{\ulcorner s_3 \urcorner} \dots p_k^{\ulcorner s_k \urcorner}$$

$2, 3, 5, \dots p_k$ je niz prostih brojeva.

$$\ulcorner v_1 \leq v_2 \Rightarrow \exists v_3 \;\; v_1 + v_3 = v_2 \urcorner = 2^{21} 3^2 5^{22} 7^5 11^9 \dots 31^{22}$$

2. Na problem postavljen u drugom pitanju odnosi se *Čerčova teza.* Odgovor je: *ne zavisi.*

3. Algoritam je konačan niz instrukcija. Instrukcija je konačan niz simbola iz najviše prebrojivog skupa. Dakle, S-izračunljivih funkcija ima najviše prebrojivo mnogo. S druge strane, aritmetičkih funkcija ima kontinuum mnogo, dakle najveći broj aritmetičkih funkcija je neizračunljiv.

4. Odgovor je: postoji skup koji je S-nabrojiv ali nije S-izračunljiv.

Primer 1. $X = \{\ulcorner \varphi \urcorner : \varphi$ je teorema Peanove aritmetike$\}$

Primer 2. Drugi primer zanimljiv je za računarstvo i povezan je sa halting problemom. Izračunljivih funkcija ima prebrojivo mnogo, neka su to $f_0, f_1, f_2, \dots$. Takodje postoji univerzalan algoritam, tj. funkcija $g(m,n)$ takva da je za sve $m, n \in N$, $g(m,n) \cong f_m(n)$. Neka $f_m(n)\downarrow$ znači "funckija f_m za ulaz (argument) n u konačno mnogo koraka daje neki izlaz". Neka je $K = \{n : g(n,n)\downarrow\}$. Tada je skup K nabrojiv ali ne i izračunljiv:

Pretpostaviom suprotno, da je karakteristična funkcija k skupa K izračunljiva. Tada je to i funkcija h definisana pomoću: $h(n)=1$ ako $k(n) = 0$ i $h(n)\uparrow$ ako $k(n)=1$; h nalazi u nizu f_n, te za neki e, $h = f_e$. Tada

$e \in K \Rightarrow g(e,e)\downarrow \Rightarrow h(e)\downarrow \Rightarrow h(e)=1 \Rightarrow k(e)=0 \Rightarrow e \notin K$,
$e \notin K \Rightarrow g(e,e)\uparrow \Rightarrow h(e)\uparrow \Rightarrow k(e)=1 \Rightarrow e \in K$,
kontradikcija (dijagonalni metod).

5. Sledeći primeri formalnih algoritamskih sistema daju odgovor na 5. pitanje:

Sistemi izračunljivosti

1. Teorija rekurzivnih funkcja $\mathcal{R}$ (K. Gödel, J. Herbrand, S. Kleene, 1930,1936).
2. Sistem Turingovih mašina $\mathcal{T}$ (A. Turing, 1936).
3. λ-račun (A. Church, 1936).
4. Funkcije definisane kanonskim deduktivnim sistemom (E. Post, 1943).
5. Algoritmi na konačnim alfabetima (A.A. Markov, 1951).
6. URM: Sitem neograničenih registarskih mašina (Shepherdson, Sturgis, 1963).
7. Programski jezici: FORTRAN, Pascal, LISP, **C**, ...

Emil Post (1944): "*Zaista, ukoliko je opšta teorija rekurzivnih funkcija formalni ekvivalent efektivne izračunljivosti, onda ova teorija u istoriji kombinatorne matematike ima mesto odmah iza pojma prirodnog broja*".

Svaki od navedenih sistema ima opšte osobine (1)-(10). Takodje, svaki od ovih sistema S definiše jednu klasu aritmetičkih funkcija $\mathcal{F}_S$:

$\mathcal{F}_{\mathcal{R}}$= skup opšte rekurzivnih funkcija.
$\mathcal{F}_{\mathcal{T}}$= skup Turing-izračunljivih funkcija.
$\mathcal{F}_{\lambda}$= skup λ-izračunljivih funkcija, ...

Ključne teoreme:

1. $\mathcal{F}_{\mathcal{T}} = \mathcal{F}_{\mathcal{R}}$ (A. Turing)
2. $\mathcal{F}_{\lambda} = \mathcal{F}_{\mathcal{R}}$ (A. Church, 1936), ...

Svaki novi formalni algoritamski sistem definisao je uvek isti skup izračunljivih funkcija, $\mathcal{F}_{\mathcal{R}}$, ili podskup od $\mathcal{F}_{\mathcal{R}}$, što ide u prilog sledećih (nedokazivih) hipoteza:

Prva Čerčova teza: *Klasa intuitivno izračunljivih aritmetičkih funkcija poklapa se sa* $\mathcal{F}_R$.

Algoritamski sistem S daje zapravo jedno *efektivno* nabrajanje S-izračunljivih funkcija. Na primer,

$\mathcal{I}_{\mathcal{R}} = \langle f_0, f_1, f_2, \dots \rangle$, niz $\mathcal{R}$-izračunljivih funkcija,
$\mathcal{I}_{\mathcal{T}} = \langle g_0, g_1, g_2, \dots \rangle$, niz $\mathcal{T}$-izračunljivih funkcija,
$\mathcal{I}_{\lambda} = \langle h_0, h_1, h_2, \dots \rangle$, niz λ-izračunljivih funkcija.

Znamo da je $\mathcal{F}_{\mathcal{T}} = \mathcal{F}_{\mathcal{R}} = \mathcal{F}_{\lambda}$, ali znamo i da je
$\mathcal{I}_{\mathcal{T}} \neq \mathcal{I}_{\mathcal{R}} \neq \mathcal{I}_{\lambda} \neq \mathcal{I}_{\mathcal{T}}$.

Pitanje Da li postoji neka odredjenija veza izmedju ovih nizova aritmetičkih funkcija?

Indeks n u potpunosti odredjuje osobine funkcije f_n u sistemu izračunljivosti $\mathcal{R}$; izmedju ostalog iz broja n može se rekonstruisati *zapis* algoritma (skup instrukcija)

u sistemu $\mathcal{R}$ koji izračunava funkciju f_n. Otuda se indeks n naziva *kôdom* funkcije f_n.

Ključne teoreme

1. *Postoji izračunljiva funkcija* φ *tako da* $g_n = f_{\varphi(n)}$,
2. *Postoji izračunljiva funkcija* ψ *tako da* $h_n = f_{\psi(n)}, \ldots$

Druga Čerčova teza: *Za bilo koji formalan ili neformalan sistem izračunljivosti* S *ima uniformna i efektivna procedura* θ *kojom se svaki skup instrukcija* P *za izračunavanje aritmetičke funkcije* f *u* S *prevodi u odgovarajući sistem instrukcija* θP *za izračunavanje funkcije* f *u standardnom sistemu* $\mathcal{R}$ (možemo uzeti da je $f = f_{\theta P}$).

Posledice formalizacije pojma algoritma

1. Izborom algoritamskog sistema, algoritam više nije neodredjen pojam, već postaje precizan isto toliko kao i pojam realnog broja, ili pojam bilo koje druge matematičke strukture. Izmedju ostalog, to znači da se može koristiti matematički aparat u izučavanju osobina ovog pojma. Drugim rečima, mogu se dokazivati teoreme koje se odnose na algoritme, a to i čini teoriju formalne izračunljivosti.

2. Naravno, i pre Gedela matematičari su imali dosta jasnu predstavu o tome šta to znači da je neki matematički zadatak efektivno rešiv. U tu svrhu, dovoljno je bilo da se navede bilo kakav efektivan postupak koji uz pomoć nekih matematičkih formula u konačno mnogo računskih koraka dovodi do rešenja. Problem je nastajao kada je trebalo utvrditi da izabrani zadatak *nema efektivnog rešenja.* U tom slučaju trebalo je dokazati matematičkim sredstvima da odgovarajući algoritam *ne postoji*, a to se nije moglo uraditi bez formalne definicije algoritma. Takav je slučaj, na primer, sa pomenutim Desetim Hilbertovim problemom u kojem se postavlja problem efektivne (algoritamske) rešivosti Diofantovskih jednačina. S obzirom da ovaj problem ima negativno rešenje, on se nije ni mogao rešiti pre pojave nove discipline – teorije formalne izračunljivosti i preciziranja pojma algoritma. To uostalom pokazuje Matijaševičevo rešenje ovog problema iz 1970.

3. Dokaz da odredjen zadatak P nije algoritamski rešiv, svodi se na dokaz da pridružena aritmetička funkcija f_P ne pripada $\mathcal{F}_{\mathcal{R}}$. Ipak, u praksi dokazi ove vrste u ogromnoj većini slučajeva idu nešto drugačijim putem, redukcijom na već spomenuti *halting problem* (**HP**), prateći sledeći obrazac: Pretpostavi se da je P algoritamski rešiv, i ukoliko se kao posledica dobije da je **HP** rešiv, imamo kontradikciju; dakle P nije algoritamski rešiv. Zato je **HP** kanonski primer algoritamski nerešivog problema, a jasno je da se **HP** ne može formulisati van nekog formalnog sistema izračunljivosti S. Podsetimo se da **HP** za sistem S ima jednostavnu formulaciju:

HP: $\quad f_n(n)\downarrow, \quad \mathcal{F}_{\mathcal{S}} = \{f_0, f_1, f_2, \ldots\}.$

Novi primeri algoritamski nerešivih problema:
Problem algebarskih identiteta (problem reči) u teoriji grupa (Novikov 1955). Ako je $u_1{=}v_1, \ldots, u_n{=}v_n$ konačan skup grupnih identiteta (primer: $a^2bc^3{=}bc^{-1}a$), da li je identitet $u{=}v$ posledica zakona teorije grupa i jednakosnih zakona.

Problem homeomorfizma. Na osnovu dve efektivno date triangulacije dveju četvoro-dimenzionalnih površi utvrditi da li su te površi homeomorfne.

Zasnivanje matematike. Pojam rekurzivne funkcije imao je ključno mesto u razvoju matematičke logike i zasnivanju matematike. Najvažniji rezultati logike su:

a. Uvodjenje preciznog simboličkog jezika u čijem okviru se svi iskazi i dokazi matematike mogu manifestovati,

b. Da nema univerzalnog algoritma pomoću kojeg bi se za proizvoljan iskaz ovog simboličkog jezika utvrdila istinitost (dokazivost) iskaza.

Potraga za rešenjem ovih zadataka datira iz mnogo ranijih vremena: setimo se *Ars Magne* Ramona Lullia i istaknimo Leibnitzovo traganje za univerzalnim simboličkim jezikom nauke (*characteristica universalis*) i za univerzalnom metodom (*calculus ratiocinator*) pomoću koje bi se manipulisalo iskazimo njegovog jezika i na taj način izvodila značenja i medjusobne relacije iskaza.

Prvi rezultat – otkriće simboličkog jezika pojavio se već u 19. i na početku 20. veka u radovima Boola, Fregea, Russela i Whiteheada. U svojoj čuvenoj *Principia mathematica* (1910) Russel i Whitehead su predstavili značajan deo matematike u takvom preciznom simboličkom ruhu. Matematičari su počeli potragu za univerzalnom procedurom koja će omogućiti da utvrdjuju istinitost iskaza u takvom simboličkom jeziku. Tokom 1920-tih veliki broj matematičara bio je aktivan u tom traganju. David Hilbert je postavio svoj čuveni program kojim bi se iz matematike u krajnjoj instanci odstranila neodredjenost i nepotpunost. Učinjen je delimičan napredak k tom cilju i neki su smatrali da je uspeh neminovan. I tada je Gödel u svom epohalnom radu *Über formal unentstcheidbare Sätze der Principia Mathematica und verwandter Systems*, Monatshefte für Mathematik und Physik, 1931, pokazao da *takve procedure nema* i usput postavio temelje teorije rekurzivnih funkcija.

Računarstvo. Izborom sistema izračunljivosti S fiksiran je niz aritmetičkih funkcija $g_0, g_1, g_2, \ldots$ izračunljivih u S. Za Sistem S možemo pretpostaviti da je neki od standardnih programskih jezika.

Pitanje. Kako ćemo dokazati da S zaista predstavlja univerzalan sistem izračunljivosti, tj. da je $\mathcal{F}_{\mathcal{S}} = \mathcal{F}_{\mathcal{R}}$? Zadatak za savremene jezike nije težak, dovoljno je da se u S simulira, na primer, sistem Turingovih mašina. Ima programskih jezika koji zapravo predstavljaju samo drugu formu nekog od poznatih algoritamskih sistema. Najpoznatiji primer ove vrste je LISP, koji predstavlja jednu implementaciju Čerčovog λ-računa. Ima i drugih jezika, uzmimo ML u kojem, na primer, postoji dokazivač za ZF sistem teorije skupova.

Čerčove teze. Prema *Prvoj Čerčovoj tezi* za svaki neformalni efektivan postupak može se u principu napisati program u S. Širenjem programskog jezika S i dodavanjem bilo kojeg broja "opcija", ne menja se polazni skup izračunljivih funkcija $\mathcal{F}_{\mathcal{S}}$. Prema *drugoj Čerčovoj tezi* za bilo koja dva programska jezika postoji algoritamski prelaz iz niza (indeksa) aritmetičkih funkcija jednog sistema u drugi. Drugim rečima,

principijelno za bilo koja dva programska jezika postoji kroskompajler - program koji prevodi programe iz prvog u drugi jezik.

Nerešivi problemi:

a. Problem da li je (ili nije) g_x konstantna funkcija.
b. Problem da li je g_x totalna funkcija.
c. Problem da li je za date x, y, z, $g_x(y) = z$.
d. Problem da li su za date x, y, g_x i g_y iste funkcije. Drugim rečima, nema algoritma kojim bi se za proizvoljne programe P i Q sistema S moglo utvrditi da li P i Q predstavljaju zapis iste funkcije.
e. Nema efektivnog postupka kojim bi se proizvoljna parcijalna izračunljiva funkcija dopunila do totalne (tj. sa domenom N) izračunljive funkcije. Drugim rečima nema algoritma pomoću kojeg bi se iz proizvoljnog programa sistema S eliminisale sve "beskonačne petlje".

Napomena. Ako je f aritmetička izračunljiva funkcija, f je parcijalna ako je $X = \text{dom(f)} \subseteq N$. Ipak, ako je i $n \in N \backslash X$ dopuštamo zapis $f(x)$, i tada kažemo da f divergira u x, što zapisujemo $f(x)\uparrow$. Ideja ove notacije je da se f zapravo izračunava u x, s tim da se proces izračunavanja nikad ne zaustavlja. To odgovara realnim situacijama kada se za neke ulazne podatke dati program P "zaglavi" u izračunavanju. Tada često kažemo da P ima "bug". Prema prethodnom primeru, nema univerzalnog algoritma za otklanjanje "bagova". Ovde vidimo da se koncept funkcije donekle razlikuje od opšteprihvaćene predstave zasnovanoj na Kantorovoj teoriji skupova, da je funkcija zapravo skup uredjenih parova (argumenata i vrednosti). Naravno, svaka izračunljiva funkcija može se smatrati kao skup uredjenih parova u prethodnom smislu. Problem je u tome što f može biti parcijalno izračunljiva, dok njen graf, domen i kodomen ne moraju biti izračunljivi (rekurzivni) skupovi.

Rekurzivne funkcije

Izračunljive funkcije imaju "algebarski opis", tj. klasa ovih funkcija može se dobiti koristeći odredjene operacije nad aritmetičkim funkcijama, polazeći od nekih jednostavnih funkcija. Pod aritmetičkom funkcijom podrazumevamo svaku funkciju $f: X \to N$, gde $X \subseteq N^k$.

Simbol $\cong$. Ako su $f(x)$ i $g(x)$ aritmetičke funkcije, onda $f(x) \cong g(x)$ znači da je vrednost $f(x)$ definisana ako i samo ako je definisana vrednost $g(x)$, i ako je $f(x)$ definisano, onda $f(x) = g(x)$.

Supstitucija. Neka su $g(\mathbf{y})$, $h_1(\mathbf{x})$, $\ldots$, $h_n(\mathbf{x})$, $\mathbf{x}, \mathbf{y} \in N$ (ovde, $\mathbf{x} = x_1, \ldots, x_m$, $\mathbf{y} = y_1, \ldots, y_n$), aritmetičke funkcije. Funkcija f dobijena je supstitucijom iz funkcija $g, h_1, \ldots, h_m$ akko za sve $\mathbf{x}$ važi

$$f(\mathbf{x}) \cong g(h_1(\mathbf{x}), \ldots, h_n(\mathbf{x}))$$

Primitivna rekurzija sa parametrima. Funkcija f dobijena je primitivnom rekurzijom iz funkcija g i h ako

$$f(\mathbf{x}, 0) \cong g(\mathbf{x}), \qquad \mathbf{x} = x_1, \dots x_{m-1}$$
$$f(\mathbf{x}, y+1) \cong h(\mathbf{x}, y, f(\mathbf{x}, y)).$$

Primitivna rekurzija bez parametara. Funkcija f dobijena je primitivnom rekurzijom iz konstante $a \in N$ i funkcije h ako: $f(0) = a$, $f(y+1) \cong h(y, f(y))$.

Minimizacija. Funkcija f dobijena je minimizacijom iz funkcije g ako za sve $\mathbf{x}$, $\mathbf{x} = x_1, \dots, x_{m-1}$, važi:

$$f(\mathbf{x}) = \text{ najmanji } y \text{ takav da } (\forall z < y)(g(\mathbf{x}, z)\downarrow \wedge g(\mathbf{x}, y) = 0)$$

Tada pišemo $f(\mathbf{x}) = \mu y(g(\mathbf{x}, y) = 0)$.

Klasa svih parcijalno rekurzivnih (p.r.) funkcija je presek svih klasa A parcijalnih aritmetičkih funkcija takvih da:

1. A sadrži konstantne, funkciju naslednik $s: x \to x+1$, i projekcijske funkcije $U_i^n: (x_1, \dots, x_n) \to x_i$,
2. A je zatvorena za operacije supstitucije, primitivne rekurzije i minimizacije.

- p.r. funkcija f je *totalna* ako za sve $\mathbf{x} \in N$, $f(\mathbf{x})\downarrow$. Za totalnu p.r. funkciju kažemo da je *opšte rekurzivna.*
- Funkcija f je *primitivno rekurzivna* ako se dobija primenom samo operacija substitucije i rekurzije polazeći od naslednik i projekcijskih funkcija.

Primer: 1. Polinomi sa koeficijentima u N i eksponencijalne funkcije (2^n,3^n,...) su primitivno rekurzivne

2. Niz prostih brojeva $2, 3, 5, 7, 11, \dots$ je primitivno rekurzivna funkcija.

3. Akermanova funkcija $a(n)$ je primer opšte rekurzivne funkcije koja nije primitivno rekurzivna. Funkcija $a(n)$ raste brže od bilo koje primitivno rekurzivne funkcije f:

$a \succ f: \quad (\exists m)(\forall n > m)\ a(n) > f(n)$

$a \succ n^2, n^3, \dots, 2^n, 3^n, \dots, 2^{2^n}, 3^{5^n} \dots, n^{n^{\cdot^{\cdot^{n}}}}, \dots$

$a(x) = A(x, x, x)$, gde je $A_z(x, y) = A(z, x, y)$ i:

$A(0, x, y) = y + x$, $A(1, x, y) = y \cdot x$, $A(2, x, y) = y^x$, ...

$A(z+1, x, y) = A_z(y, A_z(y, \dots A_z(y, y) \dots))$ ($x - 1$ puta).

4. Izračunljive funkcije iz prakse ograničene su hipereksponencijalnom funkcijom: $f \prec n^{n^{\cdot^{\cdot^{n}}}}$.