Šta je to program?

Mladen Puzić

Matematička gimnazija

23. 04. 2021.

Agenda

1. Definicija programa kroz istoriju
2. Osnove kombinatorne logike
3. Aritmetika u kombinatornoj logici

Računari i izračunavanje

- U korenu reči računar - **računati**
- Uz razvoj računara nastaje pitanje - *šta je moguće izračunati?*
- U ovom pitanju nalazi se početak računarskih nauka: u periodu od par decenija više matematičara dalo je svoju definiciju izračunljivih funkcija
- Većina definicija se sastojalo od toga da se definiše sistem objekata sa minimalnim brojem pravila, kako bismo lakše posmatrali skup funkcija koje možemo izračunati tim sistemom

Kombinatorna logika

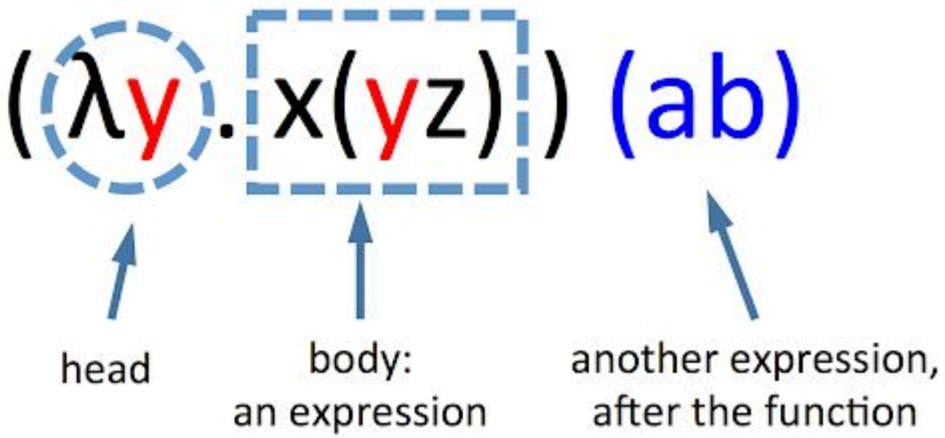
- Koncept kombinatorne logike prvi predstavio ruski logičar Mojsej Šejnfinkel još 1920. godine
- Tragičan život ga je sprečio da popularizuje ideju, ali to je uspeo američki matematičar Haskel Kari, nakon što je 1927. godine otkrio ovaj koncept
- Sistem je poznat po svojoj jednostavnosti, kao i po tome koliko je rano smišljen u odnosu na druge sisteme
- Jedini elementi alfabeta su zagrade, slova **S** i **K**, i prebrojivo promenljivih *x*, *y*, *z*...
- Dva glavna pravila koja se koriste su *S-kontrakcija* i *K-kontrakcija:*

K*xy -> x*

S*xyz -> xz(yz)*

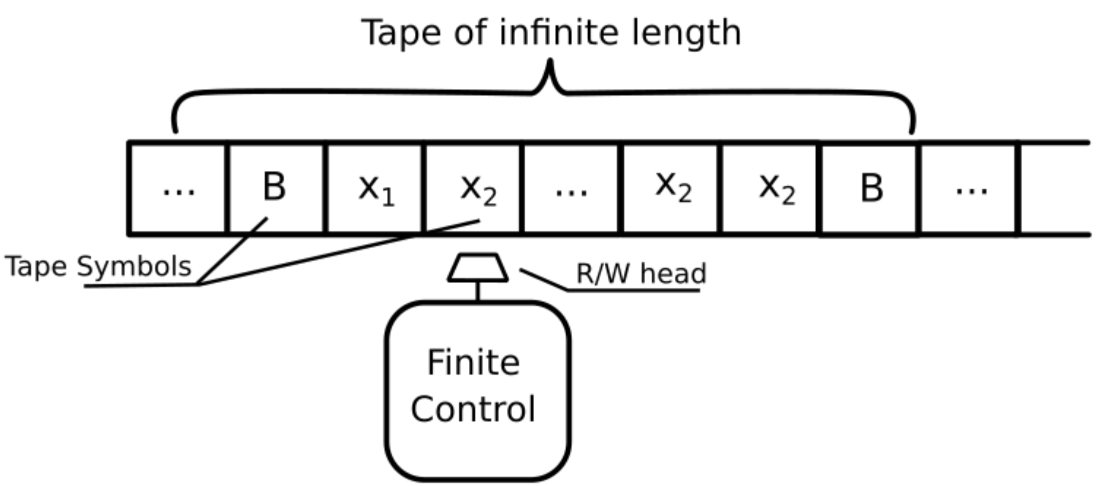
Lambda račun

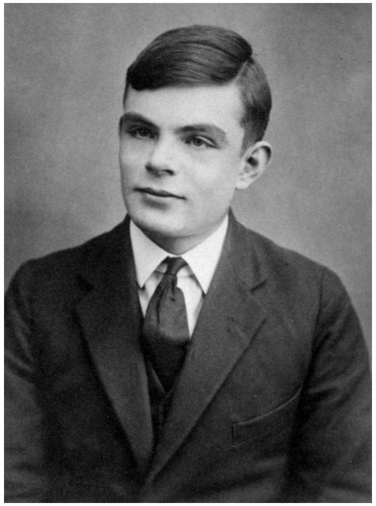
- Nastao tridesetih godina prošlog veka, smislio ga je Američki matematičar i logičar Alonzo Čerč
- Sastoji se od atomskih promenljivih, pravila građenja terama i operacija redukcija
- Blisko povezan sa kombinatornom logikom, možda najkorišćeniji sistem od navedenih



Tjuringova mašina

- Najpoznatiji termin iz ove oblasti, ovaj model smislio je engleski matematičar Alan Tjuring 1936. Godine
- Zasnovano na manipulisanju simbola na beskonačnoj *traci* (nešto nalik današnjoj kompjuterskoj memoriji) i tablici pravila
- Za svaki problem koji rešimo kompjuterom, možemo dizajnirati i Tjuringovu mašinu koja ga rešava
- Korišćen je da prvi put da negativan odgovor na *Entscheidungsproblem (decision problem)*, odnosno, da li je moguće konstruisati program koji za date aksiome i traženi zaključak proveravala da li moguće dokazati zaključak aksiomama





Unlimited Register Machine

- Objavljen 1963. godine u radu Džona Šepardsona i Hauarda Sturdžisa, kao jednostavnija alternativa Tjuringovoj mašini
- Program se sastoji od beskonačno numerisanih registara (promenljivih), koji na početku sadrže vrednost nula, i niza operacija koje se redom izvršavaju
- Imamo četiri vrste operacija:
  - *J(m, n, i):* skočiti na operaciju *i* ukoliko registri *m* i *n* sadrže istu vrednost
  - *S(n):* povećati (inkrementirati) vrednost u registru *n* za jedan
  - *T(m, n):* kopirati vrednost iz registra *m* u registar *n*
  - *Z(n):* postavlja vrednost u registru *n* na nulu
- Program se završava kada naiđe na nepostojeću operaciju (npr. kada prođe sve operacije)

1. Z(3)
2. J(3,2,100)
3. S(1)
4. S(3)
5. J(1,1,2)

Čerč-Tjuringova teza

- Svaki od navedenih sistema opisuje skup funkcija koje se mogu izračunati unutar njega
- To je navelo matematičare tog vremena da se zapitaju: *kako se ti skupovi razlikuju?*
- Odgovor je fascinantniji od bilo koje pojedinačne definicije - **sve četiri definicije opisuju isti skup funkcija!**
- Ovo je divan pokazatelj da se matematika *otkriva*, a ne *izmišlja*
- Prirodnost ove definicije dovodi do **Čerč-Tjuringove teze** - ideje da su skup funkcija koje su opisane Tjuringovom mašinom i skup funkcija koje možemo izračunati ‘papirom i olovkom’, zapravo isti skup!

Zašto nam je ovo korisno danas?

- **Lambda račun** je poslužio kao inspiracija za funkcionalne programske jezike poput *Haskella*
- **Tjuringove mašine** koriste se u teoriji složenosti, takođe najbliže imitiraju hardversku stranu računara
- **URM** je blisko povezan sa mašinskim/asemblerskim kodom, sjajno opisuje stvari koje već znamo da isprogramiramo
- **Kombinatorna logika** je važna najviše sa strane teoretskog računarstva, zbog svoje bliske veze sa lambda računom i jednostavnosti izračunavanja

Agenda

1. Definicija programa kroz istoriju
2. Osnove kombinatorne logike
3. Aritmetika u kombinatornoj logici

CL-termi

- Jezik kombinatorne logike izgrađen je koristeći zagrade, **K**, **S** i prebrojivo mnogo promenljivih v_0, v_{00}, v_{000}, ...
- **CL-termi** su objekti kojima se bavimo u kombinatornoj logici i definisani su rekurzivno:
  - **K, S** i sve promenljive su CL-termi
  - Ako su **X** i **Y** neki CL-termi, onda je i **(XY)** CL-term
  - Ništa više nije CL-term
- Ako izostavljamo zagrade, podrazumevamo da su levo-asocirane, odnosno term **XYUV** označava term **(((XY)U)V)**
- Sa **P**[**Y**/*x*] označavamo zamenu svakog pojavljivanja promenljive *x* unutar **P** sa **Y**
- CL-term koji ne sadrži promenljive nazivamo **kombinatorom**

Redukcije

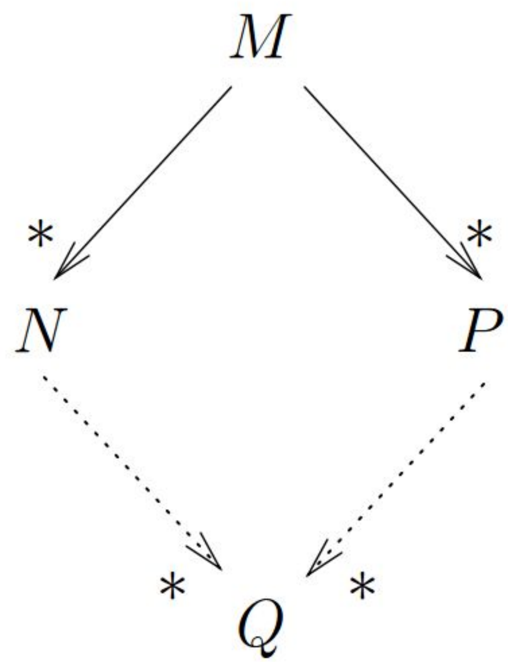
- Pravila **kontrakcije** koristimo da transformišemo jedan CL-term u drugi, odnosno da prebacimo „program'' iz jednog stanja u drugo
- Postoje samo dva pravila kontrakcije u kombinatornoj logici:
  - Zamenimo jedno pojavljivanje **KXY** sa **X**
  - Zamenimo jedno pojavljivanje **SXYZ** sa **XZ(YZ)**
- Ako kontrakcijom terma **X** možemo dobiti term **Y**, kažemo da se term **X** **kontrahuje** na term **Y**
- Ako nakon konačno mnogo kontrakcija od terma **X** možemo dobiti term **Y**, kažemo da se **X** **redukuje** na term **Y**
- Ukoliko term **X** nije moguće kontrahovati, kažemo da je term **X** **normalna forma**
- Normalna forma **Y** je normalna forma od **X** ako se **X** redukuje na **Y**

Primeri redukcija

- Podsetnik: **S***xyz* -> *xz(yz)*, **K***xy* -> *x*
- Naći kombinator **I** tako da važi **I***x* -> *x*.
  - **I** ≡ **SKK**:
    - **I***x* = **SKK***x* -> **K***x*(**K***x*) -> *x*
- Naći kombinator **B** tako da važi **B***xyz* -> *x(yz)*
  - **B** ≡ **S(KS)K**:
    - **B***xyz* = **S(KS)K***xyz* -> (**KS**)*x*(**K***x*)*yz* -> **S**(**K***x*)*yz* -> (**K***x*)*z*(*yz*) -> *x(yz)*

Čerč-Roserovo svojstvo

- Drugačije nazvano **konfluencija** ili **dijamantsko svojstvo**
- Ako se neki term **M** redukuje na dva različita terma **N** i **P**, onda se i **N** i **P** redukuju na isti term **Q**
- *Posledica:* svaki term ima najviše jednu normalnu formu



Kombinatorna kompletnost i jednačine

- **Kompletnost** kombinatorne logike podrazumeva da za svaki term **V** sa promenljivima $\{x_1, x_2, x_3, ..., x_n\}$ postoji kombinator **M** takav da:

$$\mathbf{M}\mathbf{U_1}\mathbf{U_2}\mathbf{U_3}...\mathbf{U_n} = \mathbf{V}[\mathbf{U_1}/x_1][\mathbf{U_2}/x_2][\mathbf{U_3}/x_3]...[\mathbf{U_n}/x_n]$$

- Tj. možemo naći „bilo koji'' kombinator:
  - Npr. postoji kombinator **V** takav da važi **V**$xyz = zxy$
- Takođe postoji rešenje CL jednačina i imamo jednostavan algoritam da ga nađemo:
  - Npr. jednačina $yx = x(yx)$ ima rešenje po y i možemo ga lako naći (ali zahteva malo više teorije nego što imamo vremena da ovde prođemo)

Agenda

1. Definicija programa kroz istoriju
2. Osnove kombinatorne logike
3. Aritmetika u kombinatornoj logici

Iskazna logika

- Definišimo kombinatore **t** (true) i **f** (false):

  $$\mathbf{t} \equiv \mathbf{K};$$
  $$\mathbf{f} \equiv \mathbf{KI}.$$

- Iz ovakve definicije se vidi da važi $\mathbf{t}xy = x$ i $\mathbf{f}xy = y$

- Kombinator **negacije**, odnosno kombinator **n** takav da važi:

  $$\mathbf{nt} = \mathbf{f} \text{ i } \mathbf{nf} = \mathbf{t}$$

  definišemo kao rešenje jednačine $\mathbf{n}x = x\mathbf{ft}$

- Kombinator **disjunkcije** odnosno kombinator $\vee$ takav da važi:

  $$\vee\mathbf{tt} = \vee\mathbf{tf} = \vee\mathbf{ft} = \mathbf{t} \text{ i } \vee\mathbf{ff} = \mathbf{f}$$

  definišemo kao rešenje jednačine $\vee xy = x\mathbf{t}y$

- Kombinator **konjunkcije** odnosno kombinator $\wedge$ takav da važi:

  $$\wedge\mathbf{tt} = \mathbf{t} \text{ i } \wedge\mathbf{ff} = \wedge\mathbf{ft} = \wedge\mathbf{tf} = \mathbf{f}$$

  definišemo kao rešenje jednačine $\wedge xy = xy\mathbf{f}$

Prirodni brojevi i provera nule

- Definišemo prirodne brojeve unutar CL, počevši od nule. Term koji predstavlja broj n označićemo sa $\underline{\mathbf{n}}$
- $\underline{\mathbf{0}} \equiv \mathbf{I}$
- Da bismo definisali ostale prirodne brojeve, potreban nam je kombinator **sledbenika** $\mathbf{s} \equiv \mathbf{Vf}$ (**V** je kombinator koji smo ranije definisali, $\mathbf{V}xyz = zxy$)
- $\underline{\mathbf{n+1}} \equiv \mathbf{s}\underline{\mathbf{n}}$
- Dakle, $\underline{\mathbf{1}} \equiv (\mathbf{Vf})\mathbf{I}$, $\underline{\mathbf{2}} \equiv (\mathbf{Vf})((\mathbf{Vf})\mathbf{I})$, ...

- Kombinator **prethodnika**, odnosno kombinator **P** takav da važi:

$$\mathbf{P}\ \underline{\mathbf{n+1}} = \underline{\mathbf{n}}$$

  definišemo kao rešenje jednačine $\mathbf{P}x = x\mathbf{f}$

- Kombinator **provere nule**, odnosno kombinator **Z** takav da važi:

$$\mathbf{Z}\ \underline{\mathbf{0}} = \mathbf{t} \text{ i } \mathbf{Z}\ \underline{\mathbf{n+1}} = \mathbf{f}$$

  definišemo kao rešenje jednačine $\mathbf{Z}x = x\mathbf{t}$

Sabiranje, množenje i stepenovanje

- Kombinator **sabiranja**, odnosno kombinator ⊕ takav da važi:
  $$\oplus\ \underline{\mathbf{n}}\ \underline{\mathbf{m}} = \underline{\mathbf{n+m}}$$
  definišemo kao rešenje jednačine $\oplus xy = \mathbf{Z}yx(\mathbf{s}(\oplus x(\mathbf{P}y)))$

- Kombinator **množenja**, odnosno kombinator ⊗ takav da važi:
  $$\otimes\ \underline{\mathbf{n}}\ \underline{\mathbf{m}} = \underline{\mathbf{n \cdot m}}$$
  definišemo kao rešenje jednačine $\otimes xy = \mathbf{Z}y\underline{\mathbf{0}}(\oplus(\otimes x(\mathbf{P}y))x)$

- Kombinator **stepenovanja**, odnosno kombinator **E** takav da važi:
  $$\mathbf{E}\ \underline{\mathbf{n}}\ \underline{\mathbf{m}} = \underline{\mathbf{n}}^{\mathbf{m}}$$
  definišemo kao rešenje jednačine $\mathbf{E}xy = \mathbf{Z}y\underline{\mathbf{1}}(\otimes(\mathbf{E}x(\mathbf{P}y))x)$

Dalje istraživanje:

- Knjiga *To Mock a Mockingbird* od Rejmonda Smalijana prolazi kroz kombinatornu logiku na veoma zabavan način, kroz puzle
- Moj maturski rad na temu *Kombinatorna logika i njena neodlučivost*, prolazi detaljno kroz ceo drugi deo ovog predavanja i sadrži prirodan nastavak ove priče
- J. R. Hindley, J. P. Seldin, *Lambda-calculus and combinators* - mnogo obimnije štivo o lambda računu i kombinatorima

Hvala na pažnji!

Pitanja?