

Računalnik iz domin

↓↓↓

NINA SANGAWA HMELJAK, PRIMOŽ ŠKAFAR, MAJA ŠAFARIČ
MENTOR: VID KOCIJAN

→ **Moderni računalniški centri porabijo ogromne količine električne energije, zato je v času globalnega segrevanja pomembno, da razmislimo o alternativnih načinih računanja, ki niso električno potratni. V tem članku nas bo zanimalo, ali lahko poljuben problem, ki ga lahko izračunamo s standardnim računalnikom, izračunamo s podiranjem domin.**

Z dominami bomo poskusili simulirati logične izraze in iz teh sestaviti teoretični model računalnika, imenovan Turingov stroj. Spoznali bomo idejo Church-Turingove hipoteze, ki pravi, da naj bi za vsak izračunljiv problem obstajal Turingov stroj, ki ga reši. Torej, če najdemo postopek za sestavo poljubnega Turingovega stroja iz domin, pokažemo, da lahko poljuben izračunljiv problem rešimo zgolj s podiranjem domin. Najprej bomo spoznali, kaj so logični izrazi in kaj je Turingov stroj. Iz domin bomo zgradili logične izraze, z logičnimi izrazi pa bomo Turingov stroj simulirali. Če se računalnik da simulirati z dominami, potem se ga da simulirati tudi z vsem, kar se obnaša podobno kot domine, npr. škatle za kosmiče, knjige, omare, ali pa kar nebotičniki (vsaj če si velikan v nizkokvalitetnem ameriškem filmu).

Logični izrazi

Neformalno povedano so logični izrazi funkcije logičnih spremenljivk, ki so povezane z logičnimi operacijami. Logična spremenljivka lahko zavzame le resnično vrednost »1« ali neresnično vrednost »0«. Logične izraze predstavimo s tako imenovano *resničnostno tabelo*. Vsak izraz lahko zapišemo v obliki resničnostne tabele, ki nam za vsako možno kombinacijo vrednosti vhodnih spremenljivk poda pripadajočo izhodno vrednost. Operacije, ki so nam relevantne, so:

▪ Negacija:

x	$\neg x$
0	1
1	0

▪ Konjunkcija:

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

▪ Disjunkcija:

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

▪ Strogi ali:

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Disjunktivna normalna oblika

Pokazali bomo, da lahko za vsako resničnostno tabelo sestavimo logični izraz, ki se obnaša natanko tako, kot to veli resničnostna tabela.



→ **Primer:** Recimo, da želimo dobiti logični izraz, ki bo pri vseh kombinacijah vrednosti dveh spremenljivk pravilen. Za vsako možno kombinacijo vrednosti vhodnih spremenljivk najdemo konjunkcijo, ki je resnična natanko pri teh vseh. Za vhod $x = 0$ in $y = 1$ je pripadajoča konjunkcija $\neg x \wedge y$, saj, če preberemo izraz, ta pravi, da je resničen natanko tedaj, ko x ni resničen, y pa je. Na levi strani spodnje sheme se nahaja resničnostna tabela, ki je vhod našemu postopku. Za vsako vrstico je na desni strani podan logični izraz, ki je resničen natanko pri kombinaciji vhodnih vrednosti podanih v isti vrstici.

x	y	rezultat logičnega izraza	
0	0	1	$(\neg x \wedge \neg y)$
0	1	1	$(\neg x \wedge y)$
1	0	1	$(x \wedge \neg y)$
1	1	1	$(x \wedge y)$

Nato izraze na desni povežemo z disjunkcijami, da dobimo izraz, ki je vedno pravilen:

$$\neg x \wedge \neg y) \vee (\neg x \wedge y) \vee (x \wedge \neg y) \vee (x \wedge y)$$

Primer 2: Recimo, da želimo skonstruirati logični izraz, ki se obnaša kot naslednja logična tabela:

x	y	rezultat logičnega izraza
0	0	1
0	1	1
1	0	0
1	1	0

V tem primeru potrebujemo logična izraza, ki bosta pravilna zgolj za vhode v prvih dveh vrsticah tabele. Oba logična izraza povežemo z disjunkcijo, da dobimo nov logični izraz, ki je pravilen pri natanko zgornjih dveh vseh.

x	y	rezultat logičnega izraza	
0	0	1	$(\neg x \wedge \neg y)$
0	1	1	$(\neg x \wedge y)$
1	0	0	$\neg$
1	1	0	$(\neg x \wedge \neg y) \vee (\neg x \wedge y)$

Vidimo, da bodo izrazi, zgrajeni na tak način, vedno podobne oblike, to je, več konjunkcij, povezanih z disjunkcijami. Pravimo, da so ti izrazi v disjunktivni normalni obliki. Iz primerov zgoraj je razvidno, da lahko logični izraz v disjunktivni normalni obliki zgradimo za poljubno resničnostno tabelo. To pomeni, da lahko iz negacije, disjunkcije in konjunkcije zgradimo poljuben logični izraz. Če je vhodnih spremenljivk več, bo dolžina tega izraza naraščala eksponentno. To v praksi sicer ni optimalno, vendar, ker nas zanima zgolj »Ali se da?« in ne »Ali se da učinkovito?« se zadovoljimo tudi z zelo ogromnimi formulami, če le opravijo delo.

Poln nabor operatorjev

Če lahko iz nabora operatorjev sestavimo poljuben logični izraz, pravimo, da je tak nabor operatorjev poln. Dokažimo, da lahko sestavimo poln nabor tudi z manj operatorji. Najlažji način je, da z novim naborom izrazimo nabor, za katerega že vemo, da je poln, torej $\{\neg, \wedge, \vee\}$. Negacija ($\neg$) in disjunkcija ($\vee$) sta poln nabor, saj lahko konjunkcijo $x \wedge y$ po De Morganovem zakonu izrazimo z $\neg(\neg x \vee \neg y)$

Logične operacije iz domin

Z dominami lahko modeliramo logične operacije. Logični izraz bomo zapisali kot zaporedje domin, tako da bo podiranje domin služilo kot proces računanja. Podrte domine predstavljajo logično vrednost 1, stoječe pa logično vrednost 0. Vrednost vhodnih spremenljivk vnesemo tako, da podremo pripadajoče vrste domin, nato pa počakamo, dokler se proces podiranja domin ne zaključi. Na drugi strani zaporedja podrte oziroma stoječe vrste predstavljajo izhodno vrednost spremenljivk.

Računanje v notranjosti zaporedja poteka preko logičnih operacij, ki jih sestavimo iz domin. Upora-

bljene logične operacije so predstavljene na slikah 1, 2 in 3. Na skicah se domine podirajo od spodaj navzgor.

Dokažimo, da je nabor $\{\vee, \oplus, 1\}$, ki smo ga sestavili iz domin, poln. Konstanto 1 predstavimo kot vrsto domin, ki jo na vhodu zagotovo podremo, preostali dve operaciji pa sta bili predstavljeni na slikah 1, 2 in 3.

1	y	$x \oplus 1 = \neg x$
1	0	1
1	1	0

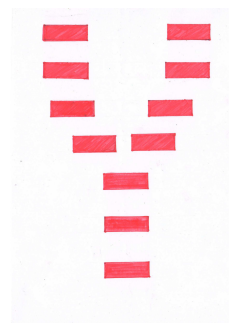
Iz tega lahko zaključimo, da iz domin lahko sestavimo tudi nabor $\{\neg, \vee\}$, za katerega smo že dokazali, da je poln. Torej lahko iz domin sestavimo poljuben logični izraz.

Turingov stroj

V tem razdelku vpeljemo skico definicij in nekaterih dokazov v zvezi s Turingovim strojem. Ker gre za matematično precej kompleksen model, se bomo natančnim definicijam ognili. Bralec, ki ga zanima točna izpeljava, lahko to najde v knjigi [1]. Turingov stroj je abstrakten model računanja, ki sledi strogim navodilom. Deli takšnega stroja so:

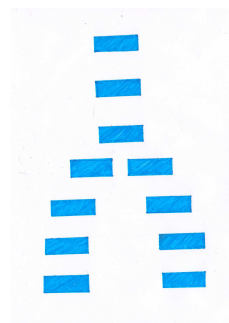
- Neskončen trak – To je spomin, na katerem so napisane 0, 1 (vhodni podatki) in prazna polja. Spomin je neomejen.
- Glava – Glava po traku bere in zapisuje podatke. V vsakem koraku se nahaja nekje na traku, kjer vsebino polja lahko prebere in prepíše, nato pa se premakne za največ eno polje levo ali desno.
- Stanja in prehodi med njimi – So stroga, natančna navodila za delovanje stroja. V vsakem koraku je stroj v enem izmed stanj. Glede na trenutno stanje stroja in znak pod glavo, prehodi natančno določajo, kaj glava zapiše na trenutno mesto, kam se premakne in v katerem stanju bo Turingov stroj v naslednjem koraku.

Alan Turing si je Turingov stroj zamislil leta 1936 in z njim matematično opredelil pojem računalnika in izračunljivosti. Turingovi stroji so po zmožnosti tega, kaj je z njimi mogoče izračunati, ekvivalentni računalniškimi programom.



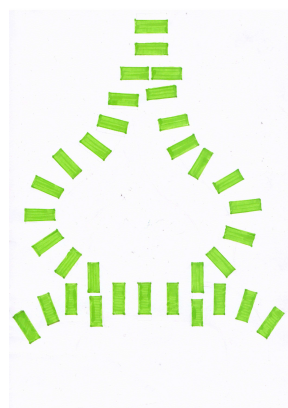
SLIKA 1.

Kopiranje: Iz enega vhoda dobimo dva izhoda (kopiranje vrednosti vhodne spremenljivke). Ta vrata uporabimo, če se neka spremenljivka v izrazu pojavi več kot enkrat.



SLIKA 2.

Disjunkcija dveh vhodnih spremenljivk



SLIKA 3.

Strogi ali dveh vhodnih spremenljivk



Turingov stroj ima končno število stanj in je v vsakem trenutku v enem izmed teh stanj, zato ga lahko predstavimo z nizom ničel in enic. Za fiksne vhodne podatke so izhodni podatki vedno enaki in ponovljivi, saj rezultat ni odvisen od zunanjih dejavnikov ali naključja.

Church-Turingova hipoteza

Church-Turingova hipoteza pravi, da za vsak izračunljiv problem obstaja Turingov stroj, ki ga reši. Torej, če Turingov stroj, ki reši problem, ne obstaja, je problem neizračunljiv tudi z najmočnejšim računalnikom. To je le hipoteza in je ne moremo dokazati, a vseeno služi kot definicija izračunljivosti. Resnična je za vse do sedaj znane praktične modele računanja.

Simulacija Turingovega stroja z logičnimi izrazi

Ker bi bil temeljit dokaz obsežen in zahteven, bomo skicirali zgolj glavne točke. Nadobudnega bralca pa vzbodbujamo, naj si več prebere v literaturi [1]. Glavne točke dokaza:

- Turingovega stroja ni mogoče predstaviti le z enim končnim logičnim izrazom. Vsak logični izraz ima fiksno velikost, vhod Turingovemu stroju pa je lahko poljubno velik. Zato popravimo naš cilj in Turingov stroj predstavimo z družino izrazov: Za vsako dolžino niza vhodnih podatkov drugačen logični izraz.
- Z logično funkcijo predstavimo i -ti korak Turingovega stroja. Vemo, da je stanje Turingovega stroja končno mnogo. Vhodnih podatkov Turingovemu stroju je bilo prav tako končno mnogo in zaradi popravljenega cilja vnaprej znano (recimo, da je bilo popisanih d elementov traku), torej je po i korakih zagotovo popisanih največ $d+i$ polj traku. Iz tega zaključimo, da je vhodnih (in izhodnih) spremenljivk naši logični funkciji končno mnogo in jo lahko sestavimo iz domin.
- Iz zgornjih dveh alinej zaključimo, da lahko rezultat enega koraka Turingovega stroja izračunamo z dominami.
- Če poznamo zgornjo mejo števila korakov Turingovega stroja, ga torej lahko predstavimo kot kompozicijo funkcij, ki izračunajo posamezne korake. Žal pa se nekateri Turingovi stroji nikoli ne

ustavijo, zato bi za simulacijo takih strojev potrebovali neskončno mnogo domin.

Praktična izvedba

Nekateri Turingovi stroji se na nekaterih vhodih ne ustavijo, ampak računajo v nedogled. Da bi tak račun simulirali, bi torej potrebovali neskončno mnogo domin, ali pa tekoči trak in robota, ki bi domine postavljali hitreje, kot se podirajo. Posledično je sestava Turingovega stroja zelo težka. Vzrok za to izhaja iz narave domin, saj se vsaka domina podre samo enkrat – realizacija povratnih zank ni mogoča. Kljub temu so preproste programe, kot je seštevanje dveh števil, že sestavili iz domin, posnetki teh eksperimentov pa so dostopni na spletnem portalu YouTube.

Vabilo na MaRS 2019

Zgornji članek je nastal kot povzetek projekta, izdelanega na matematičnem taboru MaRS 2018. Bralce vabimo, da se nam pridružijo prihodnje leto, več na <http://mars.dmfa.si/>.



Literatura

- [1] M. Sipser, *Introduction to the Theory of Computation*, Course Technology, second edition, 2006.

× × ×