Računalnik iz domin

Primož Škafar, Maja Šafarič, Nina Sangawa Hmeljak
Mentor: Vid Kocijan

Povzetek

Naša naloga je bila ugotoviti kako sestaviti računalnik (Turingov stroj) iz domin in logičnih izrazov. Ugotovili smo, da je izdelava takega računalnika teoretično mogoča, čeprav ima tak računalnik veliko omejitev.

1 Uvod

V članku bomo poskusili simulirati računalnik (Turingov stroj) z dominami. Po Church-Turingovi hipotezi 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.

Vsak korak Turingovega stroja lahko zapišemo oz. razložimo z logičnimi izrazi. Te lahko z daljšimi zaporedji domin simuliramo. Tako lahko iz domin teoretično sestavimo Turingov stroj.

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.

Najprej pa bomo spoznali, kaj so logični izrazi in kaj je Turingov stroj. Iz domin bomo zgradili logične izraze, z logičnimi izrazi pa bomo simulirali Turingov stroj.

2 Logični izrazi

Logični izrazi so funkcije logičnih spremenljivk. Poznamo logične operacije, ki so sestavni deli logičnih izrazov, kot 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

2.1 Disjunktivna normalna oblika

Vsak logični izraz lahko zapišemo v obliki resničnostne tabele. Resničnostna tabela nam za vsako možno kombinacijo vrednosti vhodnih spremenljivk poda pripadajočo izhodno vrednost. Iz resničnostne tabele lahko sestavimo logični izraz v disjunktivni normalni obliki, ki se obnaša natanko tako, kot to veli resničnostna tabela. Disjunktivno normalno obliko bomo v sledečem odstavku predstavili skozi primer.

Primer 1:

Recimo, da hočemo dobiti logični izraz, ki bo pri vseh kombinacijah vrednosti dveh spremenljivk pravilen. Za vsako možno kombinacijo vrednosti vhodnih spremenljivk najdem konjunkcijo, ki je resnična natančno pri teh vhodih. Npr. za vhod $x = 0$ in $y = 1$ je pripadajoča konjunkcija $\neg x \wedge y$.

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	$\rightarrow$	$(\neg x \wedge y)$
1	0	1		$(x \wedge \neg y)$
1	1	1		$(x \wedge y)$

Nato izraze na desni povežemo s disjunkcijami in nastane 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 sledeča 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 vhodih.

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

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

Iz primerov zgoraj je razvidno, da lahko logični izraz v obliki disjunktivne normalne oblike zgradimo za poljubno resničnostno tabelo. Torej znamo iz negacije, disjunkcije in konjunkcije zgraditi poljuben logični izraz.

2.2 Poln nabor operatorjev

Dokazali smo, da lahko z negacijo, konjunkcijo in disjunkcijo izrazimo poljuben logični izraz. Pravimo, da je tak nabor operatorjev poln.

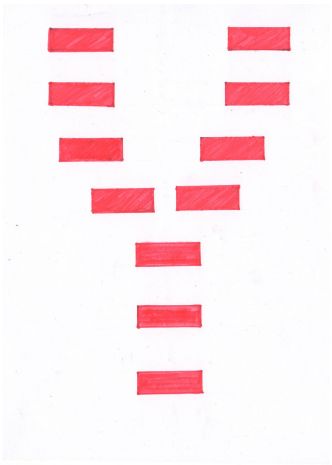
Dokažimo, da lahko sestavimo poln nabor tudi z manj operatorji. Najlažji način je, da z novim naborom izrazimo nabor, za katerega že vemo, da je poln, torej $\{\neg, \wedge, \vee\}$. Negacija ($\neg$) in disjunkcija ($\vee$) sta poln nabor, saj lahko konjunkcijo $x \wedge y$ po De Morganovem zakonu izrazimo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

2.3 Logične operacije iz domin

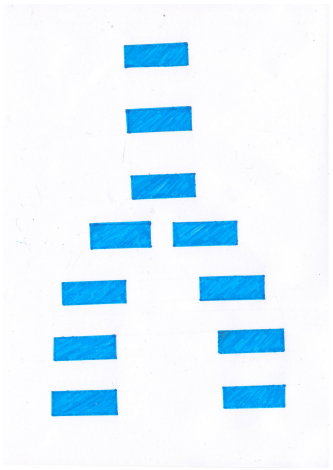
Z dominami lahko modeliramo logične operacije. Logični izraz bomo zapisali kot zaporedje domin, tako, da bo podiranje domin služilo kot proces računanja. Podrte domine predstavljajo logično vrednost 1, nepodrte pa logično vrednost 0. Vrednost vhodnih spremenljivk vnesemo tako, da podremo pripadajoče vrste domin. Podrte domine predstavljajo vrednost 1. Preostale vrste, ki jih ne podremo, predstavljajo vhodno vrednost 0. Počakamo dokler se vse domine ne podrejo. 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. Uporabljene logične operacije so predstavljene na slikah spodaj.

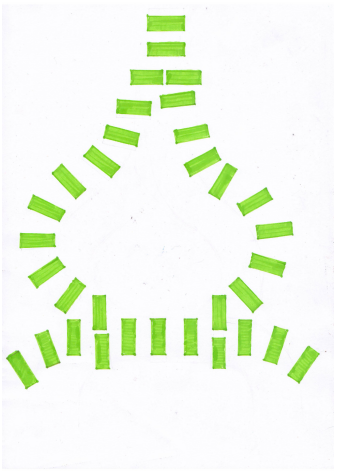
Na slikah so vhodi spodaj in izhodi zgoraj.



Slika 1: Iz enega vhoda dobimo dva izhoda (kopiranje vrednosti vhodne spremenljivke). Ta vrata uporabimo, če se neka spremenljivka pojavi v izrazu več kot enkrat.



Slika 2: Disjunkcija dveh vhodnih spremenljivk



Slika 3: Strogi ali dveh vhodnih spremenljivk

Dokažimo, da je nabor $\{\vee, \oplus, 1\}$ poln. Konstanto 1 predstavimo kot vrsto domin, ki jo na vhodu zagotovo podremo.

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

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

3 Turingov stroj

V tem razdelku vpeljemo skico definicij in nekaterih dokazov v zvezi s Turingovim strojem. Bralec, ki ga zanima točna izpeljava, lahko to najde v knjigi [1]. Turingov stroj je abstrakten model računanja. Njegovo delo posnema človeško delo in računa po strogih navodilih. Deli takšnega stroja so:

- Neskončen trak – To je spomin, na katerem so napisane 0, 1 in prazna polja. Spomin je neomejen.
- Glava – Podatke po traku piše glava. Ta se v vsakem koraku prestavi za največ eno mesto levo ali desno po traku, tam lahko bere in piše.
- Stanja in prehodi med njimi – So stroga, natančna navodila za delovanje stroja. Ta povejo kaj naj stroj napiše oz. kam se naj premakne, glede na to, kaj je napisano na trenutnem mestu pod glavo in v katerem stanju se stroj nahaja.

Alan Turing si ga je zamislil leta 1936 in z njim matematično opredelil pojem računalnika in izračunljivosti. Turingovi stroji so ekvivalentni računalniškim programom.

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 določene vhodne podatke so izhodni podatki vedno enaki in ponovljivi, saj rezulatat ni odvisen od zunanjih dejavnikov ali naključja.

3.1 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, saj je ne moremo dokazati, a vseeno služi kot definicija izračunljivosti. Resnična je za vse do sedaj znane praktične modele računanja.

3.2 Simulacija Turingovega stroja z logičnimi izrazi

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

- Turingovega stroja ni mogoče predstaviti le z enim končnim izrazom. Zato popravimo naš cilj in ga predstavimo z družino izrazov: Za vsako velikost vhoda svoj izhod.

- S popravkom definicije smo dosegli, da za posamezen program, ki ga želimo realizirati, natanko poznamo dolžino vhoda in izhoda. Iz tega lahko z veliko tehnikalijami dosežemo, da ta specifičen izračun lahko predstavimo z logično funkcijo.
- Nekateri Turingovi stroji se nikoli ne ustavijo, zato bi za simulacijo takih strojev rabili 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. Posledično je sestava Turingovega stroja zelo težka, saj je za sestavo potreben, ali zelo natančen in hiter robot, ki postavlja domine tako hitro, kot se podirajo (torej neskončno zaporedje), ali pa poznano število korakov. Vzrok za to izhaja iz narave domin, saj se vsaka domina podre samo enkrat – realizacija povratnih zank ni mogoča.

Literatura

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